University of Rhode Island

DigitalCommons@URI

Open Access Master's Theses

2020

SYNTHETIC SCENE BASED OBJECT DETECTION: CAN YOU TRAIN NEURAL NETWORKS WITH SYNTHETICALLY GENERATED DATA TO RECOGNIZE REAL OBJECTS FOR MANUFACTURING?

Gabriella M. Aiello

University of Rhode Island, gaiello@uri.edu

Follow this and additional works at: <https://digitalcommons.uri.edu/theses>

Recommended Citation

Aiello, Gabriella M., "SYNTHETIC SCENE BASED OBJECT DETECTION: CAN YOU TRAIN NEURAL NETWORKS WITH SYNTHETICALLY GENERATED DATA TO RECOGNIZE REAL OBJECTS FOR MANUFACTURING?" (2020). *Open Access Master's Theses*. Paper 1920.
<https://digitalcommons.uri.edu/theses/1920>

This Thesis is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Master's Theses by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.

SYNTHETIC SCENE BASED OBJECT DETECTION: CAN YOU TRAIN
NEURAL NETWORKS WITH SYNTHETICALLY GENERATED DATA TO
RECOGNIZE REAL OBJECTS FOR MANUFACTURING?

BY

GABRIELLA M. AIELLO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
SYSTEMS ENGINEERING

UNIVERSITY OF RHODE ISLAND

2020

MASTER OF SCIENCE THESIS
OF
GABRIELLA M. AIELLO

APPROVED:

Thesis Committee:

Major Professor Manbir Sodhi

David Taggart

Resit Sendag

Denton DeBoef

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2020

ABSTRACT

Within the manufacturing industry, Artificial Intelligence (AI) has revolutionized the way manufacturers mass produce goods and has supported a more consumer driven focus. The ability of computers to understand digital images or videos is a key sensing capability that enables connected enterprise systems. Deep learning (DL) techniques used for computer vision tasks such as object detection reliably learn features without explicitly defining said features in the training process. Evaluating deep learning techniques for object detection applications can be broken into three steps: gathering/processing data, model training/evaluation and model deployment. This research focuses on the validity of using synthetically generated data to train convolutional neural networks (CNN) to recognize objects. An object detection pipeline was created to provide an infrastructure for new models to be created using 3D computer aided design (CAD) models of objects. The pipeline was built for research purposes but can be used for application purposes beyond testing models performance. A design of experiments was created to explore significant dataset parameters as well as hyperparameters used to train deep neural networks for object detection. Some significant factors explored were dataset size, inter-class variation, learning rate, number of epochs, and type of machine learning model. Detectron2 was used to train was used to implement most object detection algorithms. Using optimized synthetically generated datasets and optimal hyperparameters, the object detection pipeline was capable of detecting objects bounding box and segmentation with average precision greater than 80%.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Dr. Sodhi at the University of Rhode Island for the encouragement to explore a technically advanced subject. The door to Prof. Sodhi's office was always open whenever I ran into a trouble spot or had a question about my research or writing. I could not have done this without his help and the help of the other members of the lab.

I would also like to thank my two additional committee members, Dr. Sendag and Dr. Taggart. Dr. Sendag investment in the project was just as important as Dr. Sodhi's and the project would not have gotten done without him.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	x
CHAPTER	
1 Introduction	1
List of References	3
2 Background	5
2.1 Object Detection in Manufacturing	5
2.2 Open Datasets and Performance Measurements	9
2.3 Synthetic Datasets and 3D Models	10
2.4 The Structure of Artificial Neural Networks	11
2.4.1 Deep Neural Network	11
2.4.2 Convolutional Neural Networks (CNNs)	12
2.4.3 Pre-trained Models	14
2.5 Object Detection Networks	14
2.5.1 R-CNN	14
2.5.2 Fast and Faster R-CNN	15
2.5.3 Mask R-CNN	16

	Page
List of References	17
3 Methodology	21
3.1 Synthetic Data Generation Pipeline	21
3.1.1 Blender	22
3.1.2 BlenderProc	29
3.2 Deep Neural Network Architecture, Training, and Evaluation . .	33
3.2.1 TensorFlow	33
3.2.2 PyTorch and Detectron2	35
List of References	37
4 Experimentation	39
4.1 Experimental setup	40
4.1.1 Datasets	40
4.1.2 Computer Hardware	43
4.1.3 Object Detection Pipeline	43
4.2 Preliminary Analysis	49
4.3 Factorial Design Analysis of Variance	55
List of References	60
5 Results	62
5.1 Preliminary Analysis Findings	62
5.1.1 Machined Object Experimentation	62
5.1.2 Lego Intraclass Variance	63
5.1.3 Lego Dataset Size Experimentation	67
5.1.4 Test Dataset Size Experimentation	70

	Page
5.2 Factorial Design Analysis of Variance Findings	70
List of References	90
6 Conclusion and Further Research	91
List of References	94
 APPENDIX	
A Blender Pipeline Python Scripts	95
A.1 Create Automated Camera Rotation Script	95
A.2 Create Pascal VOC XML Files Script	99
B TensorFlow Fixtures Object Detection Jupyter Notebook Output	108
C High and Low Variance Minitab P-Test Results	113
D Test Dataset Size Comparison of Means Results	117
E 2^k Factorial Design Results	121
F 2^k Factorial Design Minitab Results	133
G General Linear Models	148
H Two Sample T-Tests	184
I One-Way ANOVA and Descriptive Statics	189
BIBLIOGRAPHY	206

LIST OF FIGURES

Figure	Page
1 The pipeline of general CNN architecture [23]	13
2 R-CNN Architecture [27]	15
3 Fast R-CNN Architecture [28]	15
4 Faster R-CNN Architecture and RPN [29]	16
5 Mask R-CNN Architecture [30]	17
6 Blender Workflow Diagram	23
7 Blender Add-ons Preferences	24
8 Accessing Create Automated Camera Rotation	24
9 Create Automated Camera Rotation	25
10 Blender Object and Camera Rotation Setup	26
11 Blender Pipeline Output Example	27
12 Accessing Create Pascal VOC XML Files all images	28
13 Blender File Browser to Save XML Data	29
14 Blenderproc Windows Process Flow	30
15 Blenderproc Example Scene Setup	32
16 TensorFlow Pipeline Workflow Diagram	34
17 Object Detection Process from Creation of Datasets to Testing Results	36
18 Linux Based Object Detection Pipeline	44
19 COCO Evaluation Metrics [4]	47
20 IoU: Depiction of Ground Truth and Prediction [5]	48

Figure		Page
21	TensorFlow Implementation of Faster RCNN on Machined Object Fixture Model Output Example	51
22	Comparison of Segmentation Types [7]	52
23	Machined Part Detection Benchmarks	54
24	Machined Object Part Detection Benchmarks	63
25	Machined Object Part COCO Annotation	64
26	5 Objects Bounding Box Accuracy vs. Iterations by Number of Images for Training	67
27	20 Class Dataset Size Comparison	69
28	Dataset Examples	71
29	Residual Plot Bounding Box AP: Bounding box AP vs. 9 Factors and Interactions	72
30	Bounding Box Effects Graph: Bounding box AP vs. 9 Factors and Interactions	74
31	Bounding Box Results Factorial Plot Bounding box AP vs. 9 Factors and Interactions	75
32	Residual Plot Segmentation AP: Segmentation AP vs. 9 Factors and Interactions	76
33	Bounding Box Effects Graph: Bounding box AP vs. 9 Factors and Interactions	76
34	Segmentation Results Factorial Plot: Segmentation AP vs. 9 Factors and Interactions	78
35	Two-Sample T-Test Box Plots	84
36	Bounding Box Games-Howell Pairwise Comparisons	85
37	Graphic of Labeling Error	87
38	Precision vs. Time	89
39	Training Speed Effects Graph	90

40	20 Lego Class Results	94
----	---------------------------------	----

LIST OF TABLES

Table		Page
1	Comparison of Cloud API's for CV [5]	7
2	Image Recognition API's [7]	8
3	TensorFlow Model Zoo Models [6]	50
4	Detectron2 Benchmarks [8]	52
5	COCO Object Detection Baseline Models [8]	53
6	9 Factors for 2^k Factorial Design	58
7	Model Output Accuracy and Precision	63
8	Intraclass Datasets	64
9	Variance Bounding Box	65
10	Variance Segmentation	66
11	20 Class Results: LR = .02, Batch size =4, ROI =512	68
12	Bounding Box ANOVA TABLE	73
13	Segmentation ANOVA TABLE	77
14	Optimization Results	79
15	Prediction Results Comparison	81
16	Labeling Error Results	86
17	Class Comparison	87
18	Relabeled Results	88

CHAPTER 1

Introduction

Artificial intelligence (AI) enables computers to mimic human behavior and is believed by some researchers to have, or be capable of having, a human level of general intelligence. Within the manufacturing industry, AI has altered the power of the customer within manufacturing. Advances in AI related fields such as, robotics, computer vision, nanotechnology, quantum computing, biotechnology, the Internet of Things (IoT), the Industrial Internet of Things (IIoT), fifth-generation wireless technologies (5G), blockchain technologies, additive manufacturing/3D printing and fully autonomous vehicles have reshaped industry and consumer behaviors. A prime example of changes in commercialism due to AI technological advancements is the shift to ecommerce. Electronic commerce has changed the way business operate and depend on AI technologies. Artificial intelligence-enabled shopping utilizes computer vision tools to understand not only the product it is selling but personalizes online shopping experiences based on past purchase behavior.

Machine learning (ML) is a subset of AI which uses statistical methods to enable machines to improve with experiences. ML further developed into Deep Learning (DL) due to the evolution of neural networks (NN), increased processing power and the availability of big data; extremely large datasets. Neural Networks emulate the neural connectivity of the brain in order to create an algorithm. DL techniques used for computer vision tasks such as object detection, reliably understand digital images or videos by learning features without explicitly defining said features in the training process. Deep learning is computationally expensive and requires a large amount of data. Recent research, [1, 2, 3, 4, 5], focuses on reducing the amount of data used throughout the training process by increasing the qual-

ity of data and the reputability of the deep learning network. The study of cues or hyperparameters involved in transfer learning within object detection has not received much research attention. The lack of detailed methodology in machine learning practices as well as the perceived complexity of training a deep learning algorithm leaves manufacturer's reluctant to adopt AI-based technologies. There are not enough skilled software engineers within the AI industry to work with every manufacturer, but holistic machine learning pipelines as well as software tools can contribute to increased adoption of AI technology. [6] analyzed the challenges in deploying ML solutions in various applications using published reports from industry but there is a lack of academic literature that covers the challenges of deploying an entire ML pipeline. The overall object detection pipeline for this research consists of gathering/pre-processing data, modeling and model deployment. While previous work has compared the model performance using benchmark datasets, this work focused on the generation of training data as well as training predefined models. Synthetically generated data is artificially generated using computer vision rendering techniques. Pre-trained models available through PyTorch and TensorFlow serve as the base to analyze the retraining of a pre-designed network with synthetically generated data. Two different synthetic generation pipelines were used throughout experimentation: a homegrown system built with Blender, and a modular procedural pipeline called Blenderproc [7]. Additionally, multiple deep learning frameworks and object detection models were tested.

Once a framework and model were chosen, experimentation focused on testing predetermined dataset types and dataset volumes. In order to accurately track the experimental setup, a machine learning experiment tracking toolkit, Guild AI, was integrated into the pipeline. The response metrics used to measure the model's classification accuracy in detecting objects was the mean average precision (mAP)

of an object’s bounding boxes and segmentations. Other evaluation values recorded were training time, logarithmic loss, and mean absolute error. A 2^k factorial design analysis of variance was created to test training parameters related to input data but also hyperparameters used to train the convolutional neural networks. Deeper analysis of dataset testing and model tuning relied upon basic statistical concepts such as hypothesis testing to compare model results.

This research focused on the validity of using synthetically generated data to train neural networks to recognize objects. It does not focus on comparing the different methods of model deployment. Applications of the models for manufacturing will be highlighted in Chapter 2. The synthetic generation pipelines, deep neural network architecture, and training data methodology are explained in the Chapters 3 and 4. The results of the 2^k factorial design as well as results from the preliminary analysis are interpreted in Chapter 5. Conclusions and future research directions are highlighted in the Chapter 6 .

List of References

- [1] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, “On pre-trained image features and synthetic images for deep learning,” 2017.
- [2] X. Peng, B. Sun, K. Ali, and K. Saenko, “Exploring invariances in deep convolutional neural networks using synthetic images,” *CoRR*, vol. abs/1412.7122, 2014. [Online]. Available: <http://arxiv.org/abs/1412.7122>
- [3] M. Sundermeyer, Z. Marton, M. Durner, M. Brucker, and R. Triebel, “Implicit 3d orientation learning for 6d object detection from RGB images,” *CoRR*, vol. abs/1902.01275, 2019. [Online]. Available: <http://arxiv.org/abs/1902.01275>
- [4] K. Židek, P. Lazorík, J. Pitěš, and A. Hošovský, “An automated training of deep learning networks by 3d virtual models for object recognition,” *Symmetry*, vol. 11, no. 4, p. 496, Apr 2019. [Online]. Available: <http://dx.doi.org/10.3390/sym11040496>
- [5] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, “Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views,” *CoRR*, vol. abs/1505.05641, 2015. [Online]. Available: <http://arxiv.org/abs/1505.05641>

- [6] A. Paleyes, R.-G. Urma, and N. D. Lawrence, “Challenges in deploying machine learning: a survey of case studies,” 2020.
- [7] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, “Blenderproc,” *CoRR*, vol. abs/1911.01911, 2019. [Online]. Available: <http://arxiv.org/abs/1911.01911>

CHAPTER 2

Background

2.1 Object Detection in Manufacturing

Uses for object detection vary across multiple industries. Examples of object detection can be found in life sciences, manufacturing, social networking, cinematography, game design, sports tracking, and more. Object detection is a subsection of the computer vision field. Computer vision is the theory of automatically giving a computer sight and is required for computers to understand complex situations. [1] valued the annual global computer vision market at \$27.3 Billion in 2019 with a compound annual gross rate of 19% from 2020 to 2027. [2] predicted the global market for computer vision to be worth anywhere from \$17.4 billion to \$48.32 billion by 2023; as a result of the coronavirus pandemic the forecast is predicted to increase due to companies switching to digital platforms. Deloitte reports that 57% of U.S.-based survey respondents said their company had adopted computer vision [2]. Object detection is one of the core computer vision tasks that has a broad range of industrial applications such as object detection, measurement, manufacturing defect identification, identification and flaw detection.

Object detection methods fall under two categories: Machine learning and deep learning approaches. Image annotations, face detection, voice recognition, voice synthesis, and video dialog replacement are all intelligent tools based on object detection techniques, but the technological power raises ethical concerns. Most of the advances in research on DL and ML techniques have come from universities and major corporations such as Alphabet, Microsoft, IBM, Facebook, Amazon. As a result of advancements in processing power and increased data collection, the opportunity for research and development in the AI field has grown exponentially, resulting in multiple base architectures, artificial neural networks and models that

can be trained for a variety of applications [3, 4]. The mobility of the applications of object detection models has also increased as the processing power on phones and portable series to run these models increased. Modern central processing units (CPUs) and graphics processing units (GPUs) have expanded the boundaries of machine learning in computer vision. GPUs drastically cut training time due to parallel processing structures and strong floating-point capabilities. Several GPU-based convolutional neural network (CNN) model libraries have been developed to facilitate highly optimized CNN implementation on GPUs.

Besides having physical machines to train and operate object detection models, virtual machines built with Docker or similar platforms can be set up to utilize cloud-based GPU computing power such as Google Cloud Platform. Companies like Google, Microsoft, Amazon, IBM, Clarifai, and Kairos have their own commercially available cloud-based application program interfaces (API's) for computer vision that can be integrated for very specific tasks. A table outlining the features as of 2018 is depicted in Table 1 created by ActiveWizards [5]. The newest technology to speed processing power is the Tensor Processing Unit (TPU). The TPU was built by Google to accelerate deep learning tasks. Smaller versions of TPUs are physically sold and already integrated in cutting edge technologies like the google Pixel 4 smartphone but mostly utilized through googles cloud infrastructures. Google sells TPU usage by the TPU or Pod of TPU's [6].

Table 2 from Nordic APIs [7] defines API's and features for image recognition. Some systems have more pre-configured models than others, but they all have their own unique key features and applications. For example, IBM Watson has a dedicated Food Model. Amazon Rekognition has image recognition features but also movement recognition. Filestack Processing API has a built-in file management system beyond their detection capabilities. Google Cloud Vision AI offers

Comparison of Cloud APIs for CV						
	Google	Azure	Amazon Rekognition	clarifai	IBM	KAIROS
FACE DETECTION	✓	✓	✓	✓	✓	✓
FACE RECOGNITION	✗	✓	✓	✗	✗	✓
FACIAL LANDMARKS	✓	✓	✓	✗	✗	✓
FEATURE DETECTION	✓	✓	✓	✓	✗	✓
SIMILAR FACES	✗	✓	✓	✗	✗	✓
EMOTION	✓	✓	✓	✗	✗	✓
LABEL DETECTION	✓	✓	✓	✓	✓	✗
LANDMARKS	✓	✓	✗	✗	✓	✗
CELEBRITIES	✗	✓	✓	✓	✗	✗
LOGO DETECTION	✓	✗	✗	✓	✗	✗
OCR	✓	✓	✓	✗	✓	✗
NSFW	✓	✓	✓	✓	✓	✗
IMAGE ANALYSIS	✓	✓	✓	✓	✓	✗
VIDEO ANALYSIS	✓	✓	✓	✗	✗	✓ for faces
CUSTOM MODEL CREATION	✗	✓	✗	✓	✓	✗
Created by ActiveWizards						

Table 1: Comparison of Cloud API's for CV [5]

pre-trained models like Amazon Rekognition and Azure Machine Learning studio. AutoML Vision offers a solution to create custom models by training on custom images. Models are trained to automatically classify images according to labels defined by the user. The opportunity for development is basically limitless. An example implementation is for an automated inspection system that is capable of identifying locating and moving parts. These systems are also being implemented in many Augmented Reality (AR) applications. AR is becoming more prevalent in manufacturing environments to empower employees as they work. Some capabilities are to recognize objects that employees are working on or loading into different pallets. Other times AR is used to display work instructions overlaid on the object being worked on. AR applications such as PTC's Vuforia Engine 8.3 utilize neural networks in the cloud by feeding it many artificial examples of the part, which

API	FEATURES
Cloud Vision API	<ol style="list-style-type: none"> 1. Object Recognition 2. Explicit Content Detection 3. Landmark Detection 4. Object Recognition 5. Return Image Descriptions 6. Entity Identification 7. Image Matching 8. OCR Recognition
Amazon Rekognition	<ol style="list-style-type: none"> 1. Object Recognition 2. Explicit Content Detection 3. Celebrity Recognition 4. Motion Capture 5. Detect Text In Image
IBM Watson Visual Recognition	<ol style="list-style-type: none"> 1. Compatible with Machine Learning 2. Several Pre-loaded Object Identification Machine Learning Models
Microsoft Image Recognition API	<ol style="list-style-type: none"> 1. Face Detection 2. Landmark Detection 3. Celebrity Detection 4. Text Recognition 5. Information Extraction From Documents 6. Image Properties Description 7. Image Content Description and Categorization
Imagga	<ol style="list-style-type: none"> 1. Automated Image Tagging 2. Automated Image Categorization 3. Creates Analytics
Filestack Processing API	<ol style="list-style-type: none"> 1. Stores Files 2. Compresses Files 3. Converts Files 4. Integrates With File Sharing Services 5. Explicit Content Detection 6. Video Tagging 7. Image Editing
Clarifai	<ol style="list-style-type: none"> 1. Automated Image Tagging 2. Face Detection 3. Celebrity Detection 4. Demographic Analysis 5. Moderation

Table 2: Image Recognition API's [7]

create a lightweight inference AI model to run on the augmented reality application based on the customer's computer-aided designs (CAD) models [8]. Another company, c3.ai, is using a ML pipeline technology marketed to manufacturers as a tool to build and test machine learning models with prepacked templates [9]. This project has a baseline to automate the tasks around training object detection models to the point that it can be deployed for any manufacturing application. This allows manufacturers to focus on the implementation of the model: the building

of apps, development of processes, and the making of business decisions. Further explanation of the data needed to train object detection models and how object detection models work is in the following section.

2.2 Open Datasets and Performance Measurements

For many years, the principal deterrent to the implementation of AI and ML technologies was the lack of data, especially in the field of computer vision. The creation of accessible, real-world, annotated 2D images such as PASCAL VOC, ImageNet, ILSRC, MS-COCO, and Open Images fueled research and development in this field but is limited [4]. Models used for computer vision are compared for accuracy, speed and memory [10]. An exorbitant amount of data is generated and stored to research and train algorithms for computer vision algorithms, and datasets are limited by the classification of images and their dimensional descriptions [3]. Traditionally, for object detection, manually annotated real 2D images datasets were used for training.

An object's appearance in a two-dimensional image is defined by its shape, texture, surface material and physical properties. For a computer to learn from real images, people classify scenes by annotating the images with different classifiers manually or using other ML feature extraction techniques. This gives pictures context but does not identify all aspects of a scene. Pascal VOC is a large image dataset that is built upon 20 categories, with 11,530 images containing 27, 450 regions of interest (ROI) annotated objects and 6,929 segmentations[11]. The MS COCO data set is 80 object categories. The dataset has object segmentation and 5 captions per image annotations for over 200,000 images[12]. Image Net has bounding box annotations for 1,034,908 images and over 100,000 categories and subcategories [13]. All datasets are composed of real images and are intended for computer vision research. Different detection models will usually distinguish which

dataset the models were trained with.

2.3 Synthetic Datasets and 3D Models

Currently, higher dimensional datasets consisting of different data formats depth map, voxel, point cloud, mesh, etc. for 3D information 3D models, videos, and RGB-D images can be used for training [14]. RGB-D is a combination of an RGB image and corresponding depth information. Large-scale datasets with CAD, RGB-D and Polygonal surface geometry are also available, such as ModelNet, ShapeNet, BOP, and many more [15]. The Princeton ModelNet project is a collection of 127,915 3D CAD models broken into 662 Object categories and 10 categories labeled with annotated orientation [16]. The Benchmark for 6D Object Pose Estimation (BOP) is another annotation style used for estimating 6D pose. A collection of 3D object models with multiple types of datasets broken down by research paper over the challenges years is available through their website. The BOP database has RGB-D training and validation image with annotated ground truth 6D object poses and camera parameters available for download, but new datasets can be created using a Blenderproc. The 2020 challenge participants were provided 350K photo-realistic training images generated by BlenderProc4BOP [17]. The idea with synthetic data is that fewer images for training and testing will need to be generated or stored for the typical CNN. [18, 19] research show networks trained with synthetic scenes get similar and optimal results compared to networks trained with real images. [20] trained deep learning networks with 3D shape orientation models, CAD models, but have high computational and memory demands. Fields such as 3D Medical Imaging utilize this type of training but have limited input data. CAD models using simulation software can capture 2 dimensional perspectives of a 3D object or scene from a simulated camera. This process is referred to as rendering and synthetic pipelines can perform per-pixel

segmentation annotations and renderings [15]. Rendered images make it easier to create more diverse scenes efficiently compared to collecting real images using more traditional approaches. Traditional models are limited to the defined data in the image sets or real life. Annotation and classification with synthetic pipelines are easier to annotate because they happen in the rendering process and can account for noise within the artificial scene. The geometry, name, and characteristics of the object are already available from the CAD model and do not have to be manually annotated. Occlusion, clutter and illumination are controlled through these synthetic pipelines because scenes . Another benefit of using CAD models is that depth data can be calculated from the scenes [18]. [21] research studies indicate that the use of synthetic images mixed with real images has resulted in a higher mAP than models trained with only synthetic data.

New research focuses on reducing the amount of data needed for machine learning with deep learning techniques centered around not only knowing what data is needed but how the network learns that the data is needed [3]. Simulating scenes of objects with texture, shape and material as well as defined camera parameters like focal lenses, orientation, and lighting makes this research easier because it enables models to train themselves to optimize scene creation if implemented correctly. [22] training method utilized the train and forget method in order to reduce the number of images the network needs. These are only a few of the ways CNN are implemented for object detection.

2.4 The Structure of Artificial Neural Networks

2.4.1 Deep Neural Network

AI enables developers to be able to build process that process information to inform future decisions. In other words, ML tools are utilized can be utilized to make data-driven decision. With deep learning techniques computer algorithms

learn without being explicitly programmed. Deep learning is a subset of machine learning and can auto extract features from raw data to train object detection algorithms and requires less data pre-processing than other ML techniques. The difference between a normal algorithm and a machine learning algorithm is that machine learning algorithms find insights within collected data to understand the given input and output data. Conventional algorithms define data by pre-built instructions within the algorithm. Machine learning algorithms find patterns or sets of instructions to fit the data. Images are used to train feature extraction methods such as deformable part models (DPM), scale invariant transform (SIFT) descriptor, speed-up robust feature (SURF) descriptor, and Histogram of Oriented Gradient (HOG) [4].

Deep learning models based on artificial neural networks further expand upon ML methods by creating multi-layered representations of the input data [15]. These types of models learn high-level abstractions in data by detection, description and extraction [15, 23]. Layers of artificial neurons trained from the input data comprise the hidden layers of an artificial neural network. Neurons, also called perceptrons, can take multiple inputs and each perceptron in one layer, passes an activation function the next layer and readjusts the weight of every node in. Once a deep neural network has learned how to distinguish data correctly, it can then use what it has learned to make determinations about new data [15].

2.4.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks are a region proposal deep learning object detection technique. CNNs are the base for other common object detection models like R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN. CNN were modeled after the human visual cortex and structured by multilayered perceptrons. The structure of general CNN architecture as depicted in [23] is shown in Figure 1.

Figure 1: The pipeline of general CNN architecture [23]

The three main aspects of a convolutional neural network are convolutional layers, pooling layers, and fully connected layers (FCs). Filters are made up of predefined kernels of a certain pixel matrix size. Filters run across a matrix of image pixels referred to as the input feature map. The multiplication of the kernel and input feature map form an output matrix. For each convolution, each neuron in a hidden layer will compute the weighted sum of its inputs and apply bias and activate with a local non-linearity. Each neuron in the hidden layer is only seeing a patch from the original input image. CNN's form intermediate feature maps as well as feature maps per whole image throughout the process. The pooling layer computation reduce the size of the feature map with average pooling and max pooling computations. They are a strategy used to learn higher order features like shapes or specific objects within the layers of the model. The fully connected layers convert 2D feature maps into 1D feature vectors. This is the step where the model uses the vector to identify the object's classification. The combination of convolutional layers and other layering techniques are what defines a model. Deep learning is about using the layer of operations to learn the hierarchy of features. Common CNNs for image classification tasks are Alexnet, ResNet and VGG. [24] is a model with 5 convolutions and optional max pooling layers, and 3 fully connected layers. The last fully connected layer gets fed to a 1000 class SoftMax function to label objects. The neural network is trained with 1.2 million high-resolution images

from ImageNet LSVRC-2010 contest and achieved top-1 and top-5 error rates of 37.5% and 17.0%, respectively. [25] is 19 layers and architecture are based on 3x3 convolutional layers and 2x2 pooling layers. [26] is residual network made of 152 layers trained with ImageNet dataset. The model reports a 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task.

2.4.3 Pre-trained Models

It is a common practice to use transfer learning to test datasets or model capabilities by using pre-trained models as a starting point to build a new model. Pre-trained models used for transfer learning are models initialized with the weight of previously trained model that great retrained for a new dataset or application. Most of these models are trained with the MS COCO datasets or the Pascal VOC dataset. This can accelerate the learning process as well as improve the generalizability of the model. The process of refining models to modify to a specific task and datasets is called fine-tuning and requires class labels for the new training dataset. This new data is used to compute the loss functions through retraining the deep learning network, converge faster and boost performance.

2.5 Object Detection Networks

2.5.1 R-CNN

Region-based Convolutional Neural Networks (R-CNN) is an object detection model made up of a selective search method extract for CNN features. The architecture of R-CNN models is shown in Figure 2 from [27]. An image runs through the selective search region proposal network as well as a pre-trained deep CNN like ResNet. Then, the model is fine-tuned for $K + 1$ classes, and the additional class is the background. The CNN generates a feature vector that is fed to a binary SVM object classifier trained for each class independently and a bounding box

Figure 2: R-CNN Architecture [27]

offset regressor independently.

2.5.2 Fast and Faster R-CNN

Fast R-CNN, presented in [28], is built like an R-CNN model, but the main difference is that instead of extracting CNN feature vectors independently for each region proposal, the model aggregates them into one CNN forward pass over the entire image and the region proposals share this feature matrix. The architecture of Fast R-CNN model is depicted in Figure 3 . The benefit of this method is that

Figure 3: Fast R-CNN Architecture [28]

the training can update all network layers including the convolutional layers before the RoI pooling layers. The RoI pooling layer replaces the last max pooling layer of the pre-trained CNN with a RoI pooling layer. The pooling layer outputs a fixed length feature vector that is fed through the last fully connected layer to a bounding box regression model to predict classes.

Figure 4: Faster R-CNN Architecture and RPN [29]

[29] introduced a Faster R-CNN. This model composed of the region proposal network (RPN) and fast R-CNN meshed into one model with shared convolutional feature layers. The architecture of Fast R-CNN model is depicted in Figure 4. The model pre-trained a CNN network on image classification task and fine tunes the region proposal network by sliding a $n \times n$ matrix window over the convolution feature map of the entire image. The number of region proposals for each location is denoted as k anchor boxes. These proposals are then used to train the Fast R-CNN object detection model. Then, the network uses the Fast R-CNN network to initialize RPN training process then fine tunes the unique layers of the Fast R-CNN. Then it is fed through the last fully connected layer to a SoftMax estimator and a bounding box regression model to predict classes.

2.5.3 Mask R-CNN

[30] is mask R-CNN model is based on the Faster R-CNN model but a Mask fully convolutional network (FCN) predictor was added to make instance segmentation predictions. Figure 5 defines the input image fed through the network and the added instance segmentation to the network's prediction seen in the output image. This model can produce pixel-level segmentations of classes rather than just bounding boxes alone. Due to added complexity, the RoIAlign pooling layer

Figure 5: Mask R-CNN Architecture [30]

improved to fix the location misalignment caused by quantization in the RoI pooling. The model is optimized for a loss combining classification, localization and segmentation.

Further explanation of the how synthetic data is created and how CNN can be implemented for object detection are explained in the chapters 3 and 4.

List of References

- [1] X. Wu, D. Sahoo, and S. C. H. Hoi, “Recent advances in deep learning for object detection,” *Neurocomputing*, vol. 396, pp. 39–64, 2020. [Online]. Available: <https://doi.org/10.1016/j.neucom.2020.01.085>
- [2] “Computer vision market,” 2018. [Online]. Available: <https://www.marketsandmarkets.com/PressReleases/computer-vision.asp>
- [3] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015. [Online]. Available: <https://science.sciencemag.org/content/349/6245/255>
- [4] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” *CoRR*, vol. abs/1905.05055, 2019. [Online]. Available: <http://arxiv.org/abs/1905.05055>
- [5] I. Bobriakov, “Comparison of top 6 cloud apis for computer vision,” Oct 2018. [Online]. Available: <https://medium.com/activewizards-machine-learning-company/comparison-of-top-6-cloud-apis-for-computer-vision-ebf2d299be73>

- [6] Y. Wang, G. Wei, and D. Brooks, “Benchmarking tpu, gpu, and CPU platforms for deep learning,” *CoRR*, vol. abs/1907.10701, 2019. [Online]. Available: <http://arxiv.org/abs/1907.10701>
- [7] “7 best image recognition apis: Nordic apis —.” [Online]. Available: <https://nordicapis.com/7-best-image-recognition-apis/>,journal={NordicAPIs},publisher={NordicAPIsAB},author={Simpson,J.},year={2020},month={Apr}
- [8] “Object recognition,” 2020. [Online]. Available: <https://library.vuforia.com/articles/Training/Object-Recognition>
- [9] “Enterprise ai,” Oct 2020. [Online]. Available: <https://c3.ai/>
- [10] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: <http://arxiv.org/abs/1611.10012>
- [11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, June 2010.
- [12] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: <http://arxiv.org/abs/1405.0312>
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- [14] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, “Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views,” *CoRR*, vol. abs/1505.05641, 2015. [Online]. Available: <http://arxiv.org/abs/1505.05641>
- [15] T. Georgiou, Y. Liu, W. Chen, and M. S. Lew, “A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision,” *Int. J. Multim. Inf. Retr.*, vol. 9, no. 3, pp. 135–170, 2020. [Online]. Available: <https://doi.org/10.1007/s13735-019-00183-w>
- [16] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, “3d shapenets for 2.5d object recognition and next-best-view prediction,” *CoRR*, vol. abs/1406.5670, 2014. [Online]. Available: <http://arxiv.org/abs/1406.5670>
- [17] T. Hodan, M. Sundermeyer, B. Drost, Y. Labbe, E. Brachmann, F. Michel, C. Rother, and J. Matas, “Bop challenge 2020 on 6d object localization,” 2020.

- [18] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. Zakharov, H. Kosch, and J. Ernst, “Depthsynth: Real-time realistic synthetic data generation from CAD models for 2.5d recognition,” *CoRR*, vol. abs/1702.08558, 2017. [Online]. Available: <http://arxiv.org/abs/1702.08558>
- [19] K. Židek, P. Lazorík, J. Pitel, and A. Hošovský, “An automated training of deep learning networks by 3d virtual models for object recognition,” *Symmetry*, vol. 11, no. 4, p. 496, Apr 2019. [Online]. Available: <http://dx.doi.org/10.3390/sym11040496>
- [20] X. Peng, B. Sun, K. Ali, and K. Saenko, “Learning deep object detectors from 3d models,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV ’15. USA: IEEE Computer Society, 2015, p. 1278–1286. [Online]. Available: <https://doi.org/10.1109/ICCV.2015.151>
- [21] P. S. Rajpura, R. S. Hegde, and H. Bojinov, “Object detection using deep cnns trained on synthetic images,” *CoRR*, vol. abs/1706.06782, 2017. [Online]. Available: <http://arxiv.org/abs/1706.06782>
- [22] Z. Li and D. Hoiem, “Learning without forgetting,” *CoRR*, vol. abs/1606.09282, 2016. [Online]. Available: <http://arxiv.org/abs/1606.09282>
- [23] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding,” *Neurocomput.*, vol. 187, no. C, p. 27–48, Apr. 2016. [Online]. Available: <https://doi.org/10.1016/j.neucom.2015.09.116>
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: <https://doi.org/10.1145/3065386>
- [25] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv 1409.1556*, 09 2014.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [27] 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. [Online]. Available: <http://arxiv.org/abs/1311.2524>
- [28] R. B. Girshick, “Fast R-CNN,” *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: <http://arxiv.org/abs/1504.08083>
- [29] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: <http://arxiv.org/abs/1506.01497>

- [30] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06870>

CHAPTER 3

Methodology

Pre-configured models available through PyTorch and TensorFlow served as the base to test retraining a pre-trained network with synthetically generated data. Two different synthetic generation pipelines were used throughout experimentation, a homegrown system built with Blender, and a modular procedural pipeline called Blenderproc. Multiple object detection models and frameworks were tested. Once a framework and model were chosen, experimentation focused on testing predetermined dataset types and volume datasets. To accurately track the experimental setup, a machine learning experiment tracking toolkit, Guild AI, was implemented into the pipeline. The evaluation metrics used to measure the model's classification accuracy in detecting objects were bounding boxes and segmentation mean average precision. Other response values recorded were training time, logarithmic loss, and mean absolute error. A 2^k factorial design analysis of variance was created to test training parameters related to input data but also hyperparameters used to train the convolutional neural networks. A deeper analysis of dataset testing and model tuning relied upon basic statistical concepts such as hypothesis testing to compare model results.

3.1 Synthetic Data Generation Pipeline

The most time-consuming part of training a deep learning neural network is gathering accurate training data. Errors in the training setup or data labeling can cause the training to fail and result in errors within the network. A considerable amount of time of this project was focused solely on creating and verifying the synthetic data generation pipeline. Synthetic data includes digitally created images, videos, and 3D environments. A scene is a compilation of digitally created

objects, placed in space, with defined noise features such as the material properties of the surface, composition, lighting, etc. Simulation modeling techniques are used to create realistic synthetic data by manipulating lighting, texture, shape, and material as well as the camera’s focal lenses and orientation of the camera or object. The use of simulated data within the computer vision world is widely investigated. Open source rendering pipelines such as Blender, BlenderProc, Blesor, and Depthsynth can be used to create replicas of the real scenes and to render image datasets for training [1, 2] One example of synthetically generated data is from [3]. They used computer graphics from modern open-world games like Grand Theft Auto to produce semantic segmentation datasets. [4] explores a domain randomization technique that transfers by randomizing rendering to create enough variability in the simulator and thus the data. This intrinsically creates enough scenes that the real world may appear to the model as just another variation. The study found that with enough variability in the training data style, the real-world simply appears as another variation to the model. Diversity in styles has a greater effect than simulating in as realistic of an environment as possible. For this research, it was assumed that the quality of a model depended on not only the realistic quality of training images but the variety of training images as well. Two different rendering pipelines were created in the process of this research. The first pipeline was built to capture all angles of an object in space. The second pipeline was built to create randomized scenes, lighting, and camera positions.

3.1.1 Blender

An open-source, 3D computer vision software called Blender served as the base of the first synthetic data generation pipeline. This software was used for 3D modeling and object creation apart from the pipeline’s use. Blender-Python API’s were used to automate the scene by loading, rendering, and annotating images of

synthetically generated objects. It is important to note that Blender uses its own python environment. Blender’s GPU-enabled rendering engine was set to produce 600 x 480-pixel images with proper object texturing and visual effects. Settings within blender can be adjusted to change the size, visual effects, object texturing, the rendering process and many other Blender specific operations. Setting the frames to be square with higher amount of pixilation is possible but larger images take longer to render and have larger image sizes and larger annotation file sizes due to added segmentation points per object. Annotations of objects bounding boxes were generated in the form of Pascal VOC XML files for each image. A

Figure 6: Blender Workflow Diagram

simple workflow of the process is outlined in Figure 6.

The process begins with object CAD models. The CAD models are imported into Blender and two blender add-ons need to be activated to utilize Blender’s built-in functions to create an automated camera rotation around an object centered at the origin of the scene. The first script was called "Create Automated Camera Rotation" and the second was "Create Pascal VOC XML Files. A copy of the scripts is in Appendix A. These scripts were written in python and are integrated

directly into the blender interface as an add on menu as seen in Figure 7.

Figure 7: Blender Add-ons Preferences

Create Automated Camera Rotation Menu

The Create Automated Camera Rotation Menu can be used to create a camera frame to capture every angle of the object when rendering. The blender script created rotates a camera around the object vertically and takes key frames for each horizontal angle. The menu can be accessed through object commands as seen in Figure 8. The actual menu pops up at the bottom of the screen with 4

Figure 8: Accessing Create Automated Camera Rotation

settings: Distance, Horizontal Steps, Vertical Steps and Half Rotation.

Figure 9: Create Automated Camera Rotation

Figure 9 shows the blender interface when the menu pops up as well as the menu that appears. This script can also be run and set by command line augments.

The **Distance** refers to the camera's distance from the center of the world, the World origin (0,0,0). 3D CAD model of objects should be centered at the world origin by the center of mass. The distance set here determines the start camera position from the center. The camera frame should catch every part of the object when rendering.

The **Horizontal Steps** changes the number of keyframes the camera takes in one horizontal rotation. It can be changed from 1 to 360 keyframes. When the Horizontal Steps are changed to 360 the Horizontal Angle number changes to 1. This means the camera will take 360 frames (every 1 degree) for each vertical step.

The **Vertical steps** represent how many views about the z-axis the camera will create keyframes of. The Vertical angle represents the degrees at which the camera will take photos. If the Vertical Step is set to 1 the image will be taken at 180 degrees (from the top of the z-axis). The number of horizontal steps would not matter in this case because the camera is along the z-axis. To create images of the top, bottom and every angle in between, the horizontal steps should be set

to 360, the vertical steps to 180, and the automated camera rotation would render 64,442 images.

Figure 10: Blender Object and Camera Rotation Setup

The **Half Rotation** setting is meant to be used when an object is centered in the world such that it is symmetrical about the x-axis. If an object is symmetrically centered about the x-axis, one should click “Use half Rotation”. This will only render images based on the horizontal steps and vertical steps but only about 180 degrees.

After all material, background, and camera rotation settings have been properly adjusted, as seen in Figure 10, the images can be rendered using the render animation command. Images are rendered at every keyframe set in the blender. An example of the output of the pipelines is seen in Figure 11.

The keyframes start with 0 and end with the number of keyframes determined by the camera rotation. Images were saved by the “filename.backgroundname.objectcolorname_” to the file path with the frame number appended to the end. Figure 11 is labeled with proper image name examples. The key frames are set using the defined variables and utilizing Euler’s in Blender. [5]

(a) Wood1_grey_4089.jpg

(b) NikeWhite_grey_4089.jpg

(c) WhiteMarble_Black_4089.jpg

(d) Seamlessgrey_Black_4089.jpg

Figure 11: Blender Pipeline Output Example

Euler's are defined by 3 angles in radian and are used to describe the orientation of a rigid body with respect to a fixed coordinate system. Because the program is set to have the objects centered about the coordinate system it is easy to rotate the camera position, angle and focus with respect to the center of the scene.

Create Pascal VOC XML Files

Once the camera rotation settings are inputted, the rendering process is called using blenders build in Render animation function. The render animation function renders images for the active scene. Afterward, another Blender add-on, Create XML Files set, is called to generate XML files in the Pascal VOC bounding box format. The script can be accessed through the "Create Pascal VOC xml Files all images" object commands as seen in Figure 12. Pascal VOC defines the object's bounding box by x-top left, y-top left, x-bottom right, y-bottom right. XML files for each image contain information about the filename, size, and object details. The **folder** is the folder that contains the images. The **filename** is the name of the physical file that exists in the folder. The **size** contains the dimensions

Figure 12: Accessing Create Pascal VOC XML Files all images

of the image in terms of width, height, and depth. Depth of black and white images is set to 1 and for RGB depth images, it is 3. The **object details** are composed of object tags by name, pose, truncated, difficult, and bounding box. An object tag for each object in the scene is created. The name of the object is what is trying to be identified. Truncated indicates the bounding box identified is not the full shape of the object, indicating occlusion. When the object is fully visible it is set to 0 and if an object is partially visible in the image then it is set to 1. Objects known to be difficult to recognize can also be marked in this section. The final tag is the bounding box which indicates the area in the image in which the object is visible. The process of creating an annotation can vary in speed due to the amount of annotation data decided by the image. Once the "Create Pascal VOC xml Files all images" is called, a menu to set folder to save the xml files as well as set the name of the xml files to match the image names that were previously generated. This means it should also be the saved by the "filename_backgroundname_objectcolorname_" with the frame number appended to the end. An example can be seen in Figure 13. Once set, the "Create Pascal VOC xml Files all images" button needs to be clicked to begin the process but the

Figure 13: Blender File Browser to Save XML Data

window will not close until the process is done so be sure not to click out of file browser window until the process is complete.

The script used to create these files utilize information about the world coordinates and scene variable. The scene variables were the camera, resolution settings, objects. The XML data includes coordinates of the camera with respect to the origin in radian. The script filters though scenes and every object in the scene's min and max x and y values to determine if the object is fully within the key frame. If the object is fully within the key frame the x and y min and max of the objects are used to define the bounding box. X and y min and max coordinates that fall outside of the frame are replaced with the converted world to scene coordinate. Once the XML files are generated another script build into the TensorFlow Object detection pipeline creates one record of all the data. This process is further explained in 3.2.1.

3.1.2 BlenderProc

An open-source modular procedural pipeline, Blenderproc, was used to randomly generate scenes and create image datasets. Blenderproc was built on Blender to the previously stated generation pipeline but this pipeline focused on physically based rendering (PBR). PBR within object detection has been researched previously but Blenderproc has focused on providing an open-source, easily configurable pipeline to drive collaborative, innovative research in the field of computer vision. Blenderproc uses Blender-integrated cycles to create images for semantic segmentation's normal estimation and boundary detection. Though it takes longer to render PBR images, it's better to have real-world physical constraints present in

Figure 14: Blenderproc Windows Process Flow

the scene rather than to just randomly render object instances onto photo realistic images. When the scene is fully composed of physical interaction, such as view-point shading, the basic laws of physics can be rendered realistically. The pipeline set up for experimentation generates realistic images with highly identified lighting and annotated bounding box and segmentation data. Different base modules from the Blenderproc pipeline were used to create a module that randomly generates physically possible scenes of objects on a table. The experiments randomized the position and texture of the objects to be detected in the simulation, lighting, number of lights, and random noise in the background. The result is a large quantity of simulated images with randomly generated instances of objects and a variety of image conditions.

Two separate processes were created for the Blenderproc rendering pipeline. One process was created on Windows and one using a Linux system. The computer hardware used is outlined in the experimental set up. Both processes are similar

and produce the same output, but the pipeline process procedures are different due to repository differences. The main difference between pipelines was that the Linux pipeline’s rendering strategy split the generation process by 2. This was done so that the scene module would run one scene per each GPU simultaneously on the computers 2 GPU’s. Each iteration of the module runs 2 scenes, which create a random arrangement of objects and cameras and lighting. The iteration renders keyframes from each camera position. This is different from the Windows version because the Windows version only utilizes the GPU’s for certain portions of the render module. On the Linux machine, the GPUs run each process step fully for every iteration, meaning the total number of images equals the number of iterations multiplied by the number of GPUs multiplied by the number of cameras per scene. If a scene had 2 cameras set and the generation pipeline was set to run 12 iterations, a total of 48 images would be generated, 2 at a time on each GPU. The Windows pipeline process flow diagram is outlined in Figure 14. Both operating systems can run Blenderproc, but PyTorch and BlenderProc do not officially support Windows. Many of the tools commonly used for deep learning were developed on Linux.

There is a time cost to photorealistic rendering, and the pipeline was designed with speed in mind but performance first. All the scenes which the pipeline generates are pre-configured to make sure no duplicate scenes would be created in the dataset. Then, the pipeline renders the images and creates a JSON file that stores the dataset COCO annotations. The JSON format for the COCO was broken down as follows:

- Info: contains high-level information about the dataset.
- Licenses: contains a list of image licenses used in the dataset.
- Categories: contains a list of categories

Figure 15: Blenderproc Example Scene Setup

- Images: contains all the image information in the dataset without bounding box or segmentation information.
- Annotations: list of every individual object annotation from every image in the dataset.

The image and or category id need to be unique and must be sequential whole integers for the generation pipeline to work. A category can belong to a super-category and would be identified under categories. An example pipeline setup file would contain a square base named cube which the objects rest on category id labeled 1 as well as another object to be placed on top of the cube scene. If the object were a Lego piece the object would have its own Lego name, but the super-category would be a Lego. Figure 15 shows an example scene set up for a cube and an object named `cleanout_apr` centered. Both objects are centered at the origin of the blender world and have `category_id` labels. Once a scene is properly set up it can be used to run Blenderproc modules to create images and annotations. In the annotations, the COCO data format provides segmentation masks for every object instance. The bounding box in COCO is the x and y coordinates of the top left, width, and height. A Run Length Encoding (RLE) is used in the JSON generation to compute area, union or intersection efficiently. The size of the RLE representation is proportional to the number of boundary pixels of a mask.

3.2 Deep Neural Network Architecture, Training, and Evaluation

Multiple models, frameworks, and feature extractors were analyzed in order to determine which type of neural network works best when using synthetic data to retrain object detection models. Some barriers to the experimentation were that networks cannot be compared equally. Frameworks are not created equal; they have different deep learning primitives, linear operations, convolutions, and activation functions. This project focused on convolutional neural networks (CNN). This type of network works best on unstructured data such as the 2D images created from the synthetic data generation pipeline. [6, 4] proved that it is possible to produce a network with compelling performance using synthetic data. [7] froze layers responsible for feature extraction to generic layers pre-trained on real images and train only the remaining layers through synthetically generated data. Experimentation focused on different hyperparameter tuning and testing model-specific architectures for retraining pre-trained networks. Since training a neural network from scratch requires a large set of training data and training time, transfer learning was used to validate the procedural pipeline. This decreased training time and allowed increased experimentation capabilities. Pre-trained deep models for objection recognition such as (Faster-RCNN, Mask-RCNN) and image feature extractors such as Resnet50 and Resnet101 were compared using TensorFlow v2 and or the PyTorch framework.

3.2.1 TensorFlow

TensorFlow was developed by Google in 2015 and gained popularity within research due to the availability of generation documents, models, and training data. The data structure behind TensorFlow was built using tensors and other operations. PyTorch is also built around the same concept. Tensors identified multidimensional arrays or lists. The tensor structure was identified with three

Figure 16: TensorFlow Pipeline Workflow Diagram

parameters: rank, shape, and data type. Rank identified the number of dimensions of the tensor. The shape of a tensor was defined by the number of rows and columns. Data type was determined by the data fed through the system. Utilizing the NumPy library, or by converting a Python n-dimensional array into a TensorFlow tensor, TensorFlow can effectively build a model learned through image datasets. The process flow diagram depicting the machine learning pipeline developed for object detection using the TensorFlow framework is seen in Figure 16. The pipeline is built upon the Blender Synthetic Data Generation Pipeline outline. Once the blender scripts have been run to generate image and XML, the data should be stored in the Datasets directory within the pipeline directory. Data preprocessing actions such as the separation between training and validation data are set within the training_setup.py. The script generates test and train records used to train the model. Modifications to the model pipeline configuration are also set with the training_setpu.py. Once the model_main.py script is run through the command terminal, the output should be a trained model within the model's folder. The pipeline was built to utilize both GPU and CPU computing power but relies heavily on GPU usage. The TensorBoard visualization tool was used to analyze data flow graphs and training effects not only for the TensorFlow implementation but also the PyTorch implementation. The models used to test the pipeline were pre-trained on the COCO dataset but were evaluated using the PASCAL VOC 2010 detection metric. COCO evaluation metrics can be implemented with TensorFlow but were not experimented with.

3.2.2 PyTorch and Detectron2

PyTorch was developed by Facebook and publicly released in 2016. PyTorch has become a popular deep learning framework to use for object detection research. It is not as widely used or documented as TensorFlow, but its applications

have rapidly increased [8]. In November 2019, Detectron2 was released by Facebook AI. It is a PyTorch based object detection library. The Detectron2 baselines were trained with the 2017 COCO dataset and with 80 classes. To retrain the pre-trained models, a BlenderProc synthetic data generation pipeline produced properly formatted COCO annotated JSON files and images to train with. The procedural process for the pipeline to run Blenderproc and Detectron2 is different for Windows and Linux. Detectron2 does not officially support Windows but works with modifications to the Detectron2 and the Python installation supporting PyTorch. The procedural process flow for training a network with BlenderProc's

Figure 17: Object Detection Process from Creation of Datasets to Testing Results

synthetic data generation pipeline and Detectron2 is outlined in Figure 17.

The models trained with Windows used custom Jupyter Notebooks based on Detectron2 examples for object segmentation. To evaluate the network, COCO

metrics were used for bounding box detection and instance segmentation. The COCO metrics are the official detection metrics used to score the COCO competition and are like Pascal VOC metrics but have a slightly different implementation. COCO annotations report additional statistics such as mAP at IOU thresholds of .5:.95, and precision/recall statistics for small, medium, and large objects separately. This concept will be explained further in the chapter 4. The pycocotools used to evaluate the neural network caused issues with the Windows installation of Detectron2. Due to the complexity of installing dependencies with PIP through Anaconda, the Windows pipeline is not easily reproducible. A Linux based version of Detectron2 and Blenderproc pipeline was created for this reason. The Linux based object detection pipeline was used for the major study and is outlined in the experimental design, chapter 4.

List of References

- [1] D. Schraml, “Physically based synthetic image generation for machine learning: a review of pertinent literature,” in *Other Conferences*, 2019.
- [2] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. Zakharov, H. Kosch, and J. Ernst, “Depthsynth: Real-time realistic synthetic data generation from CAD models for 2.5d recognition,” *CoRR*, vol. abs/1702.08558, 2017. [Online]. Available: <http://arxiv.org/abs/1702.08558>
- [3] S. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” 08 2016.
- [4] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” *CoRR*, vol. abs/1703.06907, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06907>
- [5] “Math types utilities (mathutils),” Nov 2020. [Online]. Available: <https://docs.blender.org/api/current/mathutils.html>
- [6] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Bochoon, and S. Birchfield, “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” 2018.

- [7] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, “On pre-trained image features and synthetic images for deep learning,” 2017.
- [8] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” *CoRR*, vol. abs/1905.05055, 2019. [Online]. Available: <http://arxiv.org/abs/1905.05055>

CHAPTER 4

Experimentation

The objective of experimentation was to answer the following question: Can deep learning networks trained with synthetically generated data recognize and identify real objects? To go about answering this question, a myriad of other questions was explored in the process:

- What type of framework should be used for object detection research?
- What type of data should be used to train a deep learning model?
- What hyperparameters or experimental variables significantly affect the performance of a deep learning model to recognize objects?
- How can 3D models be used to train deep learning networks?
- Does the quality of a model depend on the realistic quality of training images?
- How many images does a network need to be retrained to recognize a singular object?
- Does domain randomization within the data affect training accuracy?
- Do different backgrounds and lighting settings affect the model's accuracy?
- Does the type of object and or the number of objects trained affect the model's accuracy?

The sensitivity of the deep learning object detection models was assessed through extensive research and experimentation with different frameworks, object detection models, and object datasets. Making sure the experimental setup was correct was

the top priority. Errors in the experimental setup or data labeling can invalidate the experiment or cause the training to fail. The importance of different factors within our training methodology was analyzed with a 2^k factorial design. The joint effects of the factors on the model accuracy helped to understand the relationship between the training data and retraining.

Once a framework and model were chosen, experimentation focused on testing predetermined dataset object types and volume datasets. The responses used to measure the model's classification accuracy and precision in detecting objects bounding boxes and segmentation were COCO evaluation metrics. Other response values recorded were training time, logarithmic loss, and mean absolute error. A factorial design analysis of variance was created to test training parameters related to input data hyperparameters. A deeper analysis of dataset testing and model tuning relied upon basic statistical concepts such as hypothesis testing to compare model results.

4.1 Experimental setup

4.1.1 Datasets

The idea of customizability of a model is important to most manufacturing applications. The types of object used to train object detection network can be trained with was important to the validity of the project. Within manufacturing, there are endless manufactured goods and assembly steps that would need to be recorded or tracked in order to take advantage of AI technology. Manufacturing knowledge is just as important as machine learning expertise in the development of intelligent automated technologies. Manufacturers have the upper hand in that most manufacturing environments are highly controlled. Parts being manufactured for sale are well tracked with various software tools and are highly documented making them ideal processes for AI implementation. There is a knowledge gap that

hinders manufacturers from utilizing machine learning for production, but with an easier system or base model to train machine learning algorithms, manufacturers can be more empowered to use customized datasets to train deep learning networks for computer vision applications.

The ability to gather enough training images from the previously stated open-source datasets is not enough for most manufacturing applications. Most uses for AI within manufacturing are for specific objects, tasks, and applications that would not be part of most benchmark datasets. Data that cannot be found from a dataset for machine learning can be created from 3D CAD files using the synthetic data generation pipeline. Engineers should have access to CAD files for anything that has been produced. Throughout the project, CAD files of objects were created or pulled from multiple sources: databases, websites, and manufacturers. 3D files that could not be obtained or made easily with 3D software were scanned with a 3D scanner. To represent the variety of manufactured parts, 3 different datasets were used for training neural networks: pipes, machined objects, and Legos.

Pipes

Pipes were used to represent a simple industrial application dataset. Piping objects come in similar colors and shapes but can be different sizes or end fittings. The symmetry and angularity of the dataset object make them interesting when assembled. This can be a complex application depending on the scene in which pipes would have to be identified. [1] uses the YOLOv3 algorithm and labels videos of defects to detect defections in sew systems. Partially occluded parts of the pipes or assemblies of pipes complicates the classification of the pipes. The pipes dataset was used for the 2^k factorial design.

Machined Objects

This dataset consists of machined metal parts from different manufacturing applications as well as 3D printed fixtures meant to hold other objects. The metal objects were often similar in design and only varied in scale and external etchings. The trays and fixtures were meant for specific objects and were also used as objects in the training dataset. The objects were used to test the variation between the scale of the objects and a model’s ability to detect the object. They were also used to test neural network’s ability to generalize textures of synthetically generated data compared to images of the real object. These objects were mostly used for the preliminary analysis of frameworks and network models. Models trained with synthetically generated images were analyzed against real images to compare which mask_rcnn network was best for retraining models.

Legos

A Lego dataset was used due to the intraclass similarities between pieces that are easily identifiable. The repeated geometric design combined with the variability in size, shape, and buildability posed an interesting basis for experimentation. Legos are also already labeled by a 4 or 5-digit design number molded on the inside of each piece. The complexity of implementing machine learning with Legos has previously been researched and implemented in various projects to sort, identify and create AR-enabled building instructions [2] [3]. The Legos were used to validate the pipeline as well as for the 2^k factorial design. For this experiment, metal, plastic, or 3D printed parts of various shapes, sizes, and uses were examined. Two different types of scanners were used for this process. A Hexagon Absolute Arm with an RS6 Laser Scanner was used with PC-Dmis software to scan parts. An EinScan Pro 2x Plus Color Pack Shining 3D scanner was used. Both scanners produce meshes that were compatible with the Blenderproc software. Many other

scanners can create meshes that would be compatible for the pipeline as well. Even phone apps produced realistic scans with proper textures. The scan of the object is important because it can affect the accuracy of the model as well as the speed in which the images render.

4.1.2 Computer Hardware

Training object detection models required advanced computing power. The experimental models were prepared and trained using an Alienware Area-51 R5 desktop powered by an Intel i7-9800X CPU, dual NVIDIA GeForce RTX 2080 GPUs, and 64gb of RAM. An Alienware Area 51m laptop powered by an Intel Core i7-9700x, a NVIDIA GeForce RTX 2080 GPU, and 64 gigabytes of RAM was also used during some areas of testing but not for the main design of experiments. Blenderproc alone requires at least a 64-bit dual core 2 GHz CPU, 4 GB RAM, OpenGL graphics TensorFlow requires a Nvidia 418 or higher and supports CUDA 10.1. Detectron2 required PyTorch version greater than 1.5 and Python version greater than 3.6.

4.1.3 Object Detection Pipeline

The object detection pipelines mentioned in the methodology was created to experiment with machine learning models. The experiments conducted with each pipeline are highlighted in the preliminary analysis. The Detectron2 Linux pipeline proved to be the easiest to implement because most of the applications used were programed in or meant to be programed in Linux. The Detectron2 Linux pipeline is shown in Figure 18 and further explanation of the pipeline is as follows.

Experiment Tracking System Setup

In order to accurately track the experimental setup, a machine learning experiment tracking toolkit, Guild AI, was implemented into the system. Guild AI

Figure 18: Linux Based Object Detection Pipeline

automated the object detection pipeline and tracked all the Guild runs. It has built in hyperparameter testing and analysis tools as well as visualization tools. For this project it was used as an experimental tracking system to record all parameters and inputs used to run Guild scripts and was used as a version control system.

The pipeline is set up to use an Anaconda environment created to support Detectron2's Linux build. Most of the data for the trials can be found within the anaconda environment folder. All the experiments run (unless otherwise deleted) are stored there. The guild file within the environment stores all runs and data as well as the metrics and per image used to test the model with written predictions saved with precision metrics.

For experiments, another branch of the directory is used to run the guild scripts. To set up a new experimental directory, create a copy of the blender_coco_template from the template folder to the current folder. This automatically creates the dependent directories as well as the guild files which store the guild operations. The guild files are what automate the pipeline to call Blenderproc and Detectron2 pipeline processes. The most important file to run the pipeline is the guild YAML file which is used to pass flags and call guild operations. Flag are used for guild operations. An example operation is generate which generates the synthetic image datasets as well as their annotations with Blenderproc. The generate operation takes a flag command to set the number of images created. The operation commands used for this experimentation were format, train, test, and run_format_train_eval. Each operation took its own set of flags and are identified in the guild YAML file. The process and guild operations are further explained by pipeline step below.

Gathering/Processing Datasets

The process of creating a machine learning model for an object begins with a need for object visualization and identification. Once the objects are identified, CAD files need to be gathered. If CAD files are not available, they need to be found online or created manually with CAD software or automatically using a 3D scanner and software. The object CAD images then need to be prepared for Blenderproc. Once a directory is set up for the experiment, a scene file with the properly labeled objects and labels should be transferred to the raw data folder. The config file was set to match the scale of the blender scene. The NIUVT environment needs to be activated for Guild AI to properly run the guild files. The generate command was called to generate scenes with the objects randomly posed on a base. These scenes were defined by a blender scene file set up in the manor outlined in the Blenderproc section of chapter 3. The collection of randomly posed object should abide by the laws of gravity, and have varying physical location, backgrounds, illuminations, occlusion.

Training Models

Once the data is generated, the format, train, and test guild operations can be called separately or at once with `run_format_train_eval`. The format operation separates the training data into 80% for train, 20% for validation and another dataset for test. The guild files are set up to default to the previously run generate, format, train when running operations. Train initiates the pretrained model and dataset along with other hyperparameters to begin training the deep learning network. The hyperparameters are not limited to the ones tested.

Once a guild train operation is called, the test operation uses the most recently run generate, format and train runs to test the ability of the trained network to recognize objects from a test dataset. The test operation can call specific generate,

format and train run if the data corresponds to the model being tested. The result of the test operation is COCO metrics for bounding box and segmentation average precision and recall. The metrics are exported as a json file and the OpenCV was used to save image prediction results.

When evaluating the model both the training data and real images were compared. Poor performance with the training data used as test data means the model has not learned properly and either the data or model need to be improved. Poor performance on real test images as the test data means the model does not generalize well. This type of over fitting mostly happened within the preliminary analysis.

COCO Evaluation Metrics

The detection evaluation metrics from Microsoft Common Objects in Context (MS COCO) were used to evaluate the performance of trained object detection models. [4] MS COCO is designed for tasks such as object detection and segmentation and is a commonly used validation set. Detectron2 maskrcnn_r_50_FPN_3x model weights are based on the COCO annotation style and are automatically generated during the rendering process.

Average Precision (AP) :	
AP	% AP at IoU=.50:.05:.95 (primary challenge metric)
AP ^{IoU=.50}	% AP at IoU=.50 (PASCAL VOC metric)
AP ^{IoU=.75}	% AP at IoU=.75 (strict metric)
AP Across Scales:	
AP ^{small}	% AP for small objects: area < 32 ²
AP ^{medium}	% AP for medium objects: 32 ² < area < 96 ²
AP ^{large}	% AP for large objects: area > 96 ²
Average Recall (AR) :	
AR ^{max=1}	% AR given 1 detection per image
AR ^{max=10}	% AR given 10 detections per image
AR ^{max=100}	% AR given 100 detections per image
AR Across Scales:	
AR ^{small}	% AR for small objects: area < 32 ²
AR ^{medium}	% AR for medium objects: 32 ² < area < 96 ²
AR ^{large}	% AR for large objects: area > 96 ²

Figure 19: COCO Evaluation Metrics [4]

The COCO website explains the evaluation metrics by evaluation task. This research used the detection evaluations metrics and is explained as follows [4]. The groups were divided by whether the object was defined as small, medium, and large. In total, COCO uses 12 main performance metrics grouped into sizes of three groups of 4 metrics. The grouping for the characterization of performance metrics is outlined in figure 19. Each object detection generated from the model will provide a bounding box (BBox) detection location and a confidence score for the predicted class. In order to measure Average Precision (AP) and Average Recall (AR), a detection needed to be determined as being True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN). Precision can be defined as the proportion of TP detected and recall is defined as the proportion of ground-truth objects successfully detected.

Figure 20: IoU: Depiction of Ground Truth and Prediction [5]

One way of determining this was to see if a given detection and ground truth of an object are the same. However, this would result in many false negatives as a result of small errors in a model's output. In many object detection and segmentation challenges, a metric called Intersection over Union (IoU) is used instead. IoU is computed by taking the area of overlap between a detection and ground truth and dividing it by the total area taken up by the detection and ground truth. Equation 1 represents this mathematically and Figure 20 from [5] visually depicts IoU.

$$IoU = \frac{target \cap prediction}{target \cup prediction} \quad (1)$$

IoU is then compared against a desired threshold to determine if a detection has occurred or not. In COCO, mAP is computed using IoU in three different ways. AP, defined by [4], is the average of 10 IoU values from 0.5 to 0.95 incremented in steps of 0.05. AP50 and AP70 are then computed by taking AP for an IoU of 0.5 and 0.75 respectively.

The AP, size metrics are computed for three size classes, small (AP_{small}), medium (AP_{medium}), and large (AP_{large}) which are defined as areas less than 32^2 , between 32^2 and 96^2 and greater than 96^2 respectively. Each size-based AP metric is calculated using the 10 averaged IoU values mentioned previously.

Average Recall has been shown to have less variation across different datasets. This metric is different from AP in that it does not perform integration on the precision-recall curve and instead takes the maximum recall.

Like AP there are two groups of AR metrics. The first computes AR based on the N largest scoring detections in an image where N is either 1, 10, or 100. These are AR1, AR10, and AR100. Like AP, AR is averaged across classes and IoU ranging from 0.5 to 0.95 in increments of 0.05. Like AP, AR is then computed for three image size groups to give ARsmall, ARmedium, and ARlarge.

4.2 Preliminary Analysis

The main objective of the preliminary analysis was to determine which features to detect and how to detect them. The first goal was to be able to detect objects and object bounding boxes from images. Bounding box detection is the ability to place a box around a target class. This proves to be a simpler task to implement than object segmentation, which identifies target classes on a per-pixel basis. The Blender generation pipeline was used to create Pascal VOC annotated datasets for the machined objects dataset. Each object was used to generate an automated camera rotation with a horizontal step of 180 and a vertical step of 36 resulting

in a total of 6,301 images. A variety of backgrounds were used to create more training data to test the influence of background color on training results. The TensorFlow2 Model Zoo [6] models were used for comparison and can be seen in Table 3 from the TensorFlow GitHub.

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
EfficientDet D0 512x512	39	33.6	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
EfficientDet D2 768x768	67	41.8	Boxes
EfficientDet D3 896x896	95	45.4	Boxes
EfficientDet D4 1024x1024	133	48.5	Boxes
EfficientDet D5 1280x1280	222	49.7	Boxes
EfficientDet D6 1280x1280	268	50.5	Boxes
EfficientDet D7 1536x1536	325	51.2	Boxes
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN ResNet152 V1 1024x1024	85	37.6	Boxes
Faster R-CNN ResNet152 V1 800x1333	101	37.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes
Faster R-CNN Inception ResNet V2 1024x1024	236	38.7	Boxes
Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks
ExtremeNet	—	—	Boxes

Table 3: TensorFlow Model Zoo Models [6]

The Faster_rcnn_resnet101_coco was retrained with TensorFlow for bounding box identification and tested on real images(see Appendix B for a copy of Jupyter

(a) Machined Object Fixtures
by Class Name

(b) Resulting Bounding Box Prediction

Figure 21: TensorFlow Implementation of Faster RCNN on Machined Object Fixture Model Output Example

notebook). An example annotated bounding box for a machined object fixture generated by a trained model can be seen in figure 21. The results of these types of classifications were satisfactory but full segmentation of the objects would be necessary for object localization needed for most use cases, so the project focus shifted to training mask object detection models for object segmentation. There are two main types of image segmentation, Semantic Segmentation and Instance Segmentation. Instance Segmentation is when objects of the same or different class are identified as different instances rather than being grouped together as a single instance, as is done in semantic segmentation. Panoptic segmentation is a combination of instance and semantic segmentation that assigns every pixel of an image a class and instance labels. The distinction between the types of segmentation are clearly distinguishable in Figure 22 from [7] panoptic segmentation research. This research focuses on instance segmentation with Mask R-CNN based models, specifically Detectron2. [8] benchmarks the training speed of its model compared to other Mask R-CNN based implementations with various networks trained with 8 NVIDIA V100 GPU's with NVLink and Python 3.7, CUDA 10.1, cuDNN 7.6.5, PyTorch 1.5, TensorFlow 1.15.0rc2, Keras 2.2.5, MxNet 1.6. The throughput of an R-CNN typically changes due to the network's dependency on the predictions

Figure 22: Comparison of Segmentation Types [7]

of the model as it trains. The benchmark metrics for Mask R-CNN implementations are shown in Table 4 [8]. Detectron2’s speed advantage in processing images through the network solidified the switch to using a PyTorch framework running Detectron2 based models for a factorial design. There are a few notable features of Detectron2. The number of object categories plus the background category is the number of objects the network is trained for. The background category ID should be 0. Detectron automatically removes the background from the dataset so the object’s category ID should not be set to 0. Detectron2 used $width = x2 - x1$ and $height = y2 - y1$. They were trained and evaluated with COCO train2017 and eval2017. Most of the models are trained with 37 epochs for 80 classes from the COCO dataset. Detectron2 COCO model baselines were compared to deter-

Implementation Model	Implementation Framework	Throughput(img/s)
Detectron2	PyTorch	62
mmdetection	PyTorch	53
maskrcnn-benchmark	PyTorch	53
tensorpak	TensorFlow	50
simpledet	mxnet	39
Detectron	Caffe2	19
matterport/Mask_RCNN	TensorFlow	14

Table 4: Detectron2 Benchmarks [8]

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	mask AP	model id
R50-C4	1x	0.584	0.110	5.2	36.8	32.2	137259246
R50-DC5	1x	0.471	0.076	6.5	38.3	34.2	137260150
R50-FPN	1x	0.261	0.043	3.4	38.6	35.2	137260431
R50-C4	3x	0.575	0.111	5.2	39.8	34.4	137849525
R50-DC5	3x	0.470	0.076	6.5	40.0	35.9	137849551
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2	137849600
R101-C4	3x	0.652	0.145	6.3	42.6	36.7	138363239
R101-DC5	3x	0.545	0.092	7.6	41.9	37.3	138363294
R101-FPN	3x	0.340	0.056	4.6	42.9	38.6	138205316

Table 5: COCO Object Detection Baseline Models [8]

mine which feature extractors and objection recognition models should be used to retrain for custom datasets for object instance segmentation and classification.

The COCO models compared with the machined object set are benchmarked in Table 5. The data used to train these models was generated with the Blender-proc pipeline and was trained with the Windows Detectron2 setup. Originally the datasets fed into the tested network were images with just one object from fixed camera angles, but in order to reduce the number of training images, images with multiple objects needed to be generated. This not only reduced training examples but it also added new dimensionality to the data. Adding more objects per image did not necessarily mean that the generation process would be faster. The images now show how multiple objects rendering and annotations per an image increased. Additional objects per scene also created occlusions within training data that could not have existed before. Now, with segmentation data, the object overlap is clearly defined. Domain randomization created more real-world scene variety. The results from this baseline comparison are in Figure 23[8]. Models image feature extractors differ by backbone combinations. FPN used a ResNet+FPN backbone with standard conv and FC heads for mask and box prediction, respectively. C4 used

Detectron2 Model Zoo Comparison

Figure 23: Machined Part Detection Benchmarks

a ResNet conv4 backbone with conv5 head [9]. DC5 (Dilated-C5) used a ResNet conv5 backbone with dilations in conv5, and standard conv and FC heads for mask and box prediction, respectively [10, 8]. Further analysis of benchmark models will be discussed in the results.

4.3 Factorial Design Analysis of Variance

In order to further understand the quality of the model and to understand the responses of the model, a factorial design analysis of variance, factorial ANOVA, was performed to test the significance of certain independent variables within the pipeline. A main effect is present when groups within a factor demonstrate a significant difference from the grand mean[11]. An interaction is present when the level of one factor is affected by the level of the other.

Variables that could possibly influence the result of the trained model are imaging conditions, objects intrinsic factors, the number of classes, number of training images, image shape, inter-class geometrical similarities or differences, intraclass variation, image degradation, and labeling accuracy. An object’s related intrinsic factors such as color, texture, material, shape and size influence the result of an object detection model. Other influencers include imaging conditions such as lighting, physical locations, randomization of camera position in training, backgrounds, illuminations, occlusion and viewing distance. Additional variables identified for training Detectron2’s maskrcnn_r_50.FPN_3x model was base network, number of epochs, learning rate, images per batch, gradient clipping, drop-out, warmup factor, ROI, learning rate decay, and early stopping.

Initial tests were run to determine which variables to experiment with. Experiments to compare:

- How training set size affects model accuracy?

- How training set size effects test time accuracy using high intraclass variance data?
- How training set size effects test time accuracy using 20 classes?

Intraclass appearance variation refers to the variation within class objects. This was tested by running intraclass experiments.

Degradation caused by image conditions such as lighting, physical location, backgrounds, illuminations, and occlusion were considered negligible for experimentation when using the Blenderproc generation pipeline because they are randomly generated within the pipeline. The variables chosen to be analyzed were Learning Rate, Learning Rate Decay, learning rate drop out iteration, batch size, Regions of Interest (ROI), number of classes, number of images per a class, epochs, number of classes, and type of dataset. These factors represent variation in the input data as well as hyperparameters for training.

Learning rate determines the weights of the network and updates the weights during the training process. The number of epochs regulates the speed with which the model learns the weights in the network. A large learning rate results in a fast learning model and requires fewer epochs but results in sub-optimal weights of the NN whereas a small learning rate make require more epochs to train and results in optimal weights. Larger learning rates results in faster convergence. Learning rate decay is the training iteration in which to begin to decay the learning rate. An initially large learning rate accelerates training or helps the network escape spurious local minima. Decaying the learning rate, lrdecay, helps the network converge to a local minimum and avoids getting stuck between local minima[12]. The learning rate decays by a certain rate at a specified number of iterations in order to prevent the network from converging and generalizing to early in the training process. Batch size, or Images per a batch, is the number of samples of

a dataset passed through a CNN in an iteration. For this experiment, mini batch gradient descent was used. Regions of interest (ROI) is a proposed region from the original image, Total number of RoI's per training

$$minibatchROI = ROIBatchSize \times BatchSize. \quad (2)$$

Epochs refer to the number of times the entire training sample dataset is passed through the learning algorithm. After each epoch, the sample in the training dataset updates the weights of the Neural Network. The number of training classes is the number of objects in the dataset. The number of images per class is the number of images to use to multiply by the number of classes to determine the size of the dataset. The number of images combined with the batch size and number of epochs makes the iterations.

$$Iterations = \frac{TotalImages \times Epochs}{Batchsize} \quad (3)$$

Iterations are used to set the other parameters within the network.

The factor levels were determined through research as well as experimentation. In order to understand the capabilities of the hardware system validation, tests with a smaller dataset training were performed on factor levels. During the training process, the loss graph determines if the model is trained accurately. Loss is a number indicating how bad the model's prediction capabilities are. The smaller the number, the better the model is at making accurate predictions.

The Detectron2 baselines are trained with the 2017 COCO dataset and 80 classes and approximately 270,000 512 x 512-pixel images. Based on a rough calculation of the specified epochs and iterations from the Detectron2 config file, for each object there should be about 1,500 images per class, $\frac{270000 \times 16}{80} = 1459$ images. The settings for the number of epochs were calculated by

$$\frac{\frac{\#classes}{80} \times 37 \times \#ofclasses}{8} \times \#ofGPU's \quad (4)$$

Factor	Low	High
Learning Rate	0.001	0.025
Learning Rate Decay	.01	.1
Learning Decline	70%	90%
ROI Batch Size	64	512
Images Per Batch	4	8
Number of Classes	5	10
Dataset	Legos	Pipes
Epochs	5	15
Number of Images Per Class	1000	1500

Table 6: 9 Factors for 2^k Factorial Design

This resulted in 3, and 12 epochs. In the original detection models, the 3x schedule were trained with 37 COCO epochs and the 1x models were trained with 12 epochs which is about a third of the total. In order to keep the ratio, the epochs were set to 5 and 15. The images per batch were determined by $\frac{16}{8} \times$ number of GPU's resulting in 4 as the lower bound. The upper bound was determined by experimenting in what the network could handle. [13, 14, 8] hyperparameters were used to determine levels of learning rate, learning rate of decay and ROI.

A 2^k factorial design was created to distinguish between main effects and potential interactions of certain factors. The main factors and their high and low settings are outlined in Table 6. The 2-level full factorial design of experiments was created with Minitab. Minitab is a statistical software package that was used to create, analyze and visualize the factorial designs and used to understand the results through graphs and ANOVA tables. In a 2-level full factorial design, each experimental factor has two level settings, a low and high setting. The experimental runs use different combinations of these factor levels to identify the training process conditions that affect the accuracy and precision of an object detection model. The number of runs necessary for a 2^k factorial design is 2^k where k is the number of factors. A 9 factor 2^k requires 512 runs but due to time constraints a 1/16 fractional

factorial design was performed with 5 replicate points totaling 160 experimental runs. Design resolutions describe how much the effects in a fractional factorial design are aliased with other effects [11]. Because a fractional factorial design used a subset of a full factorial design, it was a resolution IV design meaning that some of the 2-factor interactions were aliased with other 2-factor interactions and main effects were aliased with 3-factor interactions. For this experiment no main effects were aliased with any other main effect or 2-factor interactions. Once the design was created, all experimental runs were run using the Linux based Object detection pipeline and responses were entered into Minitab. The responses compared were the segmentation average precision and the bounding box average precision. Minitab analyzed both location and dispersion effects in the 2^k factorial design. The location model examines the relationship between the mean of the response and the factors while the dispersion model examines the relationship between the standard deviation of replicate responses and the factors [11]. Minitab fit a model to the data and generated plots and graphs to assess the effects. The final model included terms with a p-value less than .05.

Factorial plots of the main effects and interaction plots were used to determine the factor settings that optimize model results. A main effect plot is the difference in the mean response between two levels of a factor. The interaction plot shows the impact of both factors. Minitab has a built-in response optimizer to determine optimal settings of significant factors. Minitab can also be used to augment a factorial design to form a central composite design from a factorial design to create response surface regression models, surface plots and contour plots. Minitab was also used to analyze the mean responses in Analyze Factorial Design. The design of experiments optimal parameters may not transfer to models trained with larger class sizes or different object datasets. This experimentation was meant to provide

direction for future experimentation. Predictions for a 20 class Lego based dataset with optimal factors and least optimal factors were made using the fractional regression model weights. The results of the 20 class runs were compared to the predictions to determine whether the 2^k factorial design's model generalized.

List of References

- [1] X. Yin, Y. Chen, A. Bouferguene, H. Zaman, M. Al-Hussein, and L. Kurach, "A deep learning-based framework for an automated defect detection system for sewer pipes," *Automation in Construction*, vol. 109, p. 102967, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0926580519307411>
- [2] W. Yan, "Augmented reality applied to LEGO construction: Ar-based building instructions with high accuracy & precision and realistic object-hand occlusions," *CoRR*, vol. abs/1907.12549, 2019. [Online]. Available: <http://arxiv.org/abs/1907.12549>
- [3] Z. Yang, Y. Wang, C. Liu, H. Chen, C. Xu, B. Shi, C. Xu, and C. Xu, "LegoNet: Efficient convolutional neural networks with lego filters," ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 7005–7014. [Online]. Available: <http://proceedings.mlr.press/v97/yang19c.html>
- [4] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: <http://arxiv.org/abs/1405.0312>
- [5] J. JORDAN, "Evaluating image segmentation models." Dec 2018. [Online]. Available: <https://www.jeremyjordan.me/evaluating-image-segmentation-models/>
- [6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: <http://tensorflow.org/>

- [7] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic segmentation,” 2019.
- [8] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” <https://github.com/facebookresearch/detectron2>, 2019.
- [9] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: <http://arxiv.org/abs/1506.01497>
- [10] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” *CoRR*, vol. abs/1703.06211, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06211>
- [11] B. Ryan, B. L. Joiner, and J. Cryer.
- [12] J. Brownlee, *Better Deep Learning Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [14] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06870>

CHAPTER 5

Results

The results analyzed below prove that by using optimized synthetically generated datasets and optimal hyperparameters, object detection pipelines can be trained automatically with synthetically generated data. The object detection pipeline was capable of detecting objects bounding box and segmentation with average precision greater than 80% in some cases. Multiple analyses of benchmark models are discussed in the preliminary analysis findings. Intra class variance as well as Dataset Size experimentation results are highlighted in the preliminary analysis. Minitab results of the 9 factor 2^k broken down to explain statistical significance. Dataset examples and model outputs are also shown.

5.1 Preliminary Analysis Findings

The learning rate used for the preliminary analysis ranged from .001 to .025. The number of epochs was typically under 20 depending on the learning rate and the training batch size. The dataset size was typically over 1,000 images and increased depending on the number and type of objects being trained. For 10 objects (some being of similar geometries), over 6,000 images generated the most accurate results.

5.1.1 Machined Object Experimentation

The mask_rcnn_R_50_C4_3x was determined to be the best model to retrain when using randomly generated scenes of machined objects in space. The way the model generated these images randomly oriented objects in a similar way to the Blenderproc pipeline used for the main experimentation. However, the objects where not arranged on the table but in space. Example images generated to train

Positive Predictive Value: Precision	76.98%	True Negative Rate	85.31%
True Positive Rate: Recall	79.37%	False Positive Rate: Fall Out	14.69%
Accuracy	81.67%	False Negative Rate: Miss Rate	20.63%
F1 Score	76.28%	False Discovery Rate	23.02%

Table 7: Model Output Accuracy and Precision

the 3-class machined object detection network are seen in Figure 24.

Figure 24: Machined Object Part Detection Benchmarks

Another model trained with the mask_rcnn_R_50_C4_3x model to detect these machined object parts: air bearings, turbine blade, hip joint, and pallet tray for the bearings. In total, 7,000 images were generated for the 8 objects and were tested with 31 real images. The model accurately identifies multiple objects within the image, even with many bearings containing high levels of similarities seen in Figure 25. Immediately following are details of the model performance in terms of mean average precision and recall.

5.1.2 Lego Intraclass Variance

When using parts with apparent intrinsic differences, the model’s recognition performance was expected to be higher than objects with similar features. This theory was tested with the machined objects and then again with the Lego dataset. Two sets of objects were compared, one with high intraclass variance and one with low intraclass variance. Variance was determined subjectively between Lego parts. The two datasets are described in Table 8. The difference between the parts of the

Figure 25: Machined Object Part COCO Annotation

high intraclass objects is highly distinguishable while the low variance parts have similar geometric features of different quantities.

High Intraclass Variance			Low Intraclass Variance		
Lego ID	Class Name		Lego ID	Class Name	
44126	slope_c6x2		60581	panel_ss_hs
30361	brick_r_robotbody		3002	brick2x3
92290	minifigure_weapon_trident		3021	plate2x3
3035	plate4x8		3710	plate1x4
50745	vehicle_mudguard		3005	brick1x1

Table 8: Intraclass Datasets

It was expected that the model would distinguish objects of different shapes, the high variance dataset, to have a higher average precision and recall. However, it turned out that there was no statistical difference between the two datasets COCO metrics. Statistically, the bounding boxes means for the low and high variance dataset are not different. But the average recall and average precision for small object area predictions are significantly higher for the high variance dataset.

High Intraclass Variance Bounding Box Results				
Average Precision (AP) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.874
Average Precision (AP) @[IoU=0.50	area= all	maxDets=100]	=	0.927
Average Precision (AP) @[IoU=0.75	area= all	maxDets=100]	=	0.914
Average Precision (AP) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.271
Average Precision (AP) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.874
Average Precision (AP) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.934
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 1]	=	0.891
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 10]	=	0.893
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.893
Average Recall (AR) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.301
Average Recall (AR) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.900
Average Recall (AR) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.950
Low Intraclass Variance Bounding Box Results				
Average Precision (AP) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.871
Average Precision (AP) @[IoU=0.50	area= all	maxDets=100]	=	0.904
Average Precision (AP) @[IoU=0.75	area= all	maxDets=100]	=	0.902
Average Precision (AP) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.170
Average Precision (AP) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.871
Average Precision (AP) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.928
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 1]	=	0.888
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 10]	=	0.891
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.891
Average Recall (AR) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.188
Average Recall (AR) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.898
Average Recall (AR) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.941

Table 9: Variance Bounding Box

The object area is defined by the number of pixels in the object mask generated for the segmentation task [1]. It should be noted that some of object instances in scenes are so small that they are often hard to distinguish even by the human eye. The model does perform poorer for small object instances. It was expected for the network to identify object sets of varying geometric shape better than similar shape. The difference is greater than 10%, but the predictions are still relatively low. The predictions metrics for the high and low bounding box detection can be seen in Table 9. The average segmentations, however, are not statistically different and are in Table 10. The Minitab output can be found in Appendix C.

High Intraclass Variance Segmentation Results				
Average Precision (AP) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.840
Average Precision (AP) @[IoU=0.50	area= all	maxDets=100]	=	0.927
Average Precision (AP) @[IoU=0.75	area= all	maxDets=100]	=	0.908
Average Precision (AP) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.224
Average Precision (AP) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.832
Average Precision (AP) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.917
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 1]	=	0.860
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 10]	=	0.861
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.861
Average Recall (AR) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.282
Average Recall (AR) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.863
Average Recall (AR) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.925
Low Intraclass Variance Segmentation Results				
Average Precision (AP) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.876
Average Precision (AP) @[IoU=0.50	area= all	maxDets=100]	=	0.904
Average Precision (AP) @[IoU=0.75	area= all	maxDets=100]	=	0.899
Average Precision (AP) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.139
Average Precision (AP) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.882
Average Precision (AP) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.934
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 1]	=	0.890
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets= 10]	=	0.893
Average Recall (AR) @[IoU=0.50:0.95	area= all	maxDets=100]	=	0.893
Average Recall (AR) @[IoU=0.50:0.95	area= small	maxDets=100]	=	0.184
Average Recall (AR) @[IoU=0.50:0.95	area=medium	maxDets=100]	=	0.904
Average Recall (AR) @[IoU=0.50:0.95	area= large	maxDets=100]	=	0.941

Table 10: Variance Segmentation

5.1.3 Lego Dataset Size Experimentation

In another preliminary test, the training set size was tested using the low and high parameters. Four different experiments were run with a data set size of 2k, 4k, 8k and 16k images of 5 objects. The model was trained with 32k iterations with 4 images per batch and learning rate of .02. The epoch size depended on the dataset size. For the 2000 images 64 epochs were used. For 4000 images, 32 epochs were used. For 8000 images 16 epochs were used and for 16000 images 8 epochs were used. Increasing train set size consistently increased accuracy in the long run. The graph 26 depicts the accuracy vs. iterations per number of training images.

Figure 26: 5 Objects Bounding Box Accuracy vs. Iterations by Number of Images for Training

The number of images per class were 400, 800, 1,600, 3,200. Since the number of iterations was not changed per dataset size, the number of times a network passes through all the images decreased as the number of images increased. This means that when the network was fed 2,000 images for 32,000 iterations with 4 images per

a batch, the network passed through all 2,000 images 64 times. When there were 16,000 images, the network saw each one 8 times throughout the whole training process. The average precision increases 1% for each increase in data size and reduction in epoch size.

A 20 class Lego set was used to run the same four runs with dataset sizes of 2k, 4k, 8k and 16k, each being trained for 32k iterations. The number of classes was 4 times the original test size and the platform area which the objects randomly generate on also increased by a factor of 4.

Figure 27 compares how training set size affects training accuracy by dataset size for bounding box detection and segmentation with the 20-class dataset by number of images in the dataset. The average precision loss and training parameters are outlined in Table 11.

Dataset Size	Images per class	Epochs	bbox/AP	segm/AP	Total Loss
2000	100	64	77.695	76.213	0.30
4000	200	32	80.576	78.856	.37
8000	400	16	81.677	80.380	0.41
1600	800	8	83.631	82.022	0.45

Table 11: 20 Class Results: LR = .02, Batch size =4, ROI =512

Compared to the 5-class test, the average precision dropped when training with 20 classes. The big difference between the two training set size tests was that for each class in the 20-class dataset, there were 100, 200, 400, 800 images per class used. The x number of images per class doesn't mean that there were only explicitly x number of instances per object. Most images were composed of multiple objects and every image had the base cube. The validation accuracy during training is much smoother compared to the 5-class test. This is most likely due to the increased number of classes decreasing the impact of variable performance on a per class basis. Average precision is defined as a mean across per class average precision, so objects which produce poor predictions in a dataset

(a) Bounding Box Accuracy vs. Iterations

(b) Segmentation Accuracy vs. Iterations

Figure 27: 20 Class Dataset Size Comparison

with a smaller number of classes would have more of an effect than a dataset with a larger number of classes. Another possibility is that the added objects created more occlusions and object interactions within the training data, smoothing out the training results during training.

5.1.4 Test Dataset Size Experimentation

Test dataset time has limitations within the object detection pipeline. The number of test images that can be processed with a trained model is dependent on the number of instances and predictions per image, as well as the number of images. There was a problem where the desktop would run out of storage due to CUDA memory capacity limits. A test was run to see whether the amount of testing images used to test the network had a significant effect on the average precision results. If one uses over 1000 images with the test model, the program will most likely crash when running 8 items per batch. Test dataset sizes of 250, 500, and 700 were compared to see if there was a statistical difference between the average precision results. There was no statistical difference between the average means of the different amounts of training images. The statistical results for these trials are in Appendix D.

5.2 Factorial Design Analysis of Variance Findings

The 9 factor 2^k factorial design took about 1 month to run 160 experiments. The total training time to run the 160 trials was 19 days, 1 hour 26 minutes. It took about 12 hours to generate 18,000 images with 5 objects and 18 hours to generate 18,000 images with 10 classes. To generate all the images necessary to run the trials would take about 4 days. Dataset examples from each dataset used for the factorial design are in Figure 28. Due to errors in the initial scene used to generate the 10-pipe dataset, multiple generations of the 10-pipe dataset were performed and analyzed. Data used for runs requiring the 10 pipes was multiplied by the percentage change between the correct trial runs and mean of the wrong generations per experiment. This was done to save time. Rather than re-doing 40 incorrect runs, 8 runs had to be redone. Deeper explanation of the 10-pipe data correction follows the factorial experimentation. Appendix E is the experimental

(a) 10 Lego Dataset

(b) 10 Pipe Dataset

(c) 5 Lego Dataset

(d) 5 Pipe Dataset

Figure 28: Dataset Examples

log for all the trials performed for the 2^9 factorial design . The incorrect bounding box and segmentation AP value used for this calculation are in 2 columns following the generate column. The average of the repeated trials is stored in the last two columns. This data was transferred to Minitab and a model was fit to the factorial design. The regression equations that produced the best models were analyzed with test bounding box average precision, test segmentation average precision and training speed.

The factorial regression model comparing test bounding box average precision vs. 9 Factors and their interactions produced a model with an adjusted R-squared value of 99.62%. The residual plots from the model are seen in Figure 29 indicates the residuals are normally distributed and there are no trends or patterns indicating the model is adequate. There was a statistically significant association between bounding box average precision and all the main factors. The two-way interaction that was insignificant was learning rate times learning rate of decline. The only 3-

Figure 29: **Residual Plot Bounding Box AP:** Bounding box AP vs. 9 Factors and Interactions

way interaction that was significant was between Learning rate, ROI and number of images per class. The full regression results from Minitab for the bounding box results, including the regression equation, are in Appendix F. The Analysis of Variance for the bounding box results is in table 12.

The Factors with the largest effect on the model are: Learning rate of decay x Learning Rate Decline, Number of Classes, Learning Rate x ROI, ROI, Epoch, and Learning Rate. The Pareto Chart of the Standardized Effects is seen in Figure 30. .

The coefficient of the term in the regression equation represents the change in the mean response for a one-unit change of the factor term. If the sign of the coefficient is negative, as the factor increases, the average precision decreases. If the coefficient is positive, as the term increases, the average precision increases. The factorial plots in Figure 31 evaluate the relationships between the response and the variables. The main effects plot on the top of the figure shows the relationship to the individual variable, while the interaction plot shows the relationship between

Analysis of Variance: Bounding box AP vs. 9 Factors and Interactions					
Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	31	4157.45	134.11	1357.82	0.000
Linear	9	2254.57	250.51	2536.28	0.000
LR	1	157.07	157.07	1590.24	0.000
LRD	1	103.63	103.63	1049.24	0.000
LR Decline	1	0.73	0.73	7.36	0.008
Batch	1	25.42	25.42	257.35	0.000
ROI	1	453.48	453.48	4591.31	0.000
NumImagesperClass	1	32.62	32.62	330.29	0.000
NumClasses	1	1028.56	1028.56	10413.76	0.000
Dataset	1	83.72	83.72	847.60	0.000
epoch	1	369.33	369.33	3739.35	0.000
2-Way Interactions	21	1899.13	90.43	915.61	0.000
LR*LRD	1	39.48	39.48	399.76	0.000
LR*LR Decline	1	0.06	0.06	0.57	0.451
LR*Batch	1	133.82	133.82	1354.89	0.000
LR*ROI	1	460.75	460.75	4664.91	0.000
LR*NumImagesperClass	1	6.08	6.08	61.60	0.000
LR*NumClasses	1	3.19	3.19	32.28	0.000
LR*Dataset	1	1.29	1.29	13.01	0.000
LR*epoch	1	52.78	52.78	534.33	0.000
LRD*LR Decline	1	1051.98	1051.98	10650.81	0.000
LRD*Batch	1	12.12	12.12	122.71	0.000
LRD*RoI	1	11.32	11.32	114.61	0.000
LRD*Dataset	1	19.83	19.83	200.82	0.000
LRD*epoch	1	2.17	2.17	22.00	0.000
LR Decline*Batch	1	27.44	27.44	277.84	0.000
LR Decline*RoI	1	5.08	5.08	51.40	0.000
LR Decline*epoch	1	16.33	16.33	165.29	0.000
Batch*RoI	1	12.65	12.65	128.08	0.000
ROI*NumImagesperClass	1	0.51	0.51	5.15	0.025
ROI*NumClasses	1	0.78	0.78	7.89	0.006
ROI*Dataset	1	3.69	3.69	37.33	0.000
ROI*epoch	1	37.78	37.78	382.52	0.000
3-Way Interactions	1	3.76	3.76	38.03	0.000
LR*ROI*NumImagesperClass	1	3.76	3.76	38.03	0.000
Error	128	12.64	0.10		
Total	159	4170.09			

Table 12: Bounding Box ANOVA TABLE

Figure 30: **Bounding Box Effects Graph:** Bounding box AP vs. 9 Factors and Interactions

2 variables setting to determine the optimal variable values . This was used to determine optimal parameters to use when training a network.

The factorial regression model comparing test segmentation average precision vs.9 Factors and interactions produced a model with an adjuster R-squared value of 99.46%. The residual plots from the model are seen in Figure 32 indicate the residuals are normally distributed and there are no trends or patterns indicating the model is adequate. There was a statistically significant association between segmentation average precision and all the main factors. The two-way interactions that were insignificant were learning rate times learning rate of decline, learning rate times number of classes, ROI times images per class and ROI times number of classes. The full regression results from Minitab for the bounding box results, including the regression equation, are in Appendix F. The Analysis of Variance for the segmentation results is in table 13.

The Factors with the largest effect on the model are: Learning rate of decay times Learning Rate Decline, Number of Classes, type of dataset, epoch, Learning

Factorial Plots for test -evaluation ap for bbox

Figure 31: **Bounding Box Results Factorial Plot** Bounding box AP vs. 9 Factors and Interactions

Figure 32: **Residual Plot Segmentation AP:** Segmentation AP vs. 9 Factors and Interactions

Rate times ROI, ROI. The Pareto Chart of the Standardized Effects is seen in Figure 33. . The coefficient of the term in the regression equation represents

Figure 33: **Bounding Box Effects Graph:** Bounding box AP vs. 9 Factors and Interactions

the change in the mean response for a one-unit change of the factor term. If the coefficient is positive, as the term increases, average precision increases. The

Analysis of Variance: Segmentation vs. 9 Factors and Interactions					
Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	31	3229.36	104.17	1181.40	0.000
Linear	9	1686.73	187.41	2125.43	0.000
LR	1	0.36	0.36	4.06	0.046
LRD	1	36.15	36.15	410.01	0.000
LR Decline	1	1.29	1.29	14.63	0.000
Batch	1	1.76	1.76	19.96	0.000
RoI	1	154.39	154.39	1750.92	0.000
NumImagesperClass	1	21.74	21.74	246.53	0.000
NumClasses	1	830.93	830.93	9423.39	0.000
Dataset	1	441.11	441.11	5002.57	0.000
epoch	1	199.00	199.00	2256.77	0.000
2-Way Interactions	21	1540.98	73.38	832.19	0.000
LR*LRD	1	8.29	8.29	94.04	0.000
LR*LR Decline	1	0.03	0.03	0.32	0.570
LR*Batch	1	50.93	50.93	577.56	0.000
LR*RoI	1	176.04	176.04	1996.45	0.000
LR*NumImagesperClass	1	7.38	7.38	83.64	0.000
LR*NumClasses	1	0.28	0.28	3.14	0.079
LR*Dataset	1	3.03	3.03	34.31	0.000
LR*epoch	1	11.38	11.38	129.02	0.000
LRD*LR Decline	1	1213.73	1213.73	13764.64	0.000
LRD*Batch	1	19.22	19.22	217.93	0.000
LRD*RoI	1	1.54	1.54	17.46	0.000
LRD*Dataset	1	2.03	2.03	23.03	0.000
LRD*epoch	1	0.97	0.97	11.02	0.001
LR Decline*Batch	1	24.95	24.95	282.96	0.000
LR Decline*RoI	1	4.09	4.09	46.41	0.000
LR Decline*epoch	1	2.70	2.70	30.58	0.000
Batch*RoI	1	1.46	1.46	16.55	0.000
RoI*NumImagesperClass	1	0.25	0.25	2.84	0.095
RoI*NumClasses	1	0.09	0.09	0.99	0.322
RoI*Dataset	1	2.12	2.12	24.07	0.000
RoI*epoch	1	10.49	10.49	118.94	0.000
3-Way Interactions	1	1.64	1.64	18.61	0.000
LR*RoI*NumImagesperClass	1	1.64	1.64	18.61	0.000
Error	128	11.29	0.09		
Total	159	3240.64			

Table 13: Segmentation ANOVA TABLE

factorial Plots in Figure 34 evaluate the relationships between the response and the variables. The main effects plot on the top of the figure shows the relationship to the individual variable while the interaction plot shows the relationship between

Factorial Plots for test -evaluation for segm ap

Figure 34: **Segmentation Results Factorial Plot:** Segmentation AP vs. 9 Factors and Interactions

one variable and the fitted average precision depends on the second variable. This was used to determine the optimal parameters to use when training a network.

Minitab also has a built-in response optimization tools that produced the same optimal results as analyzing the factor plots. The first result of the optimization for bounding box and segmentation models is a learning rate of .025, learning rate of decay of .1, learning rate decline of .7, batch size of 8, ROI of 512, 15 epochs, 1500 images per a class, 5 classes and using the pipe dataset. The average precision of bounding boxes at these parameters was 96.002 and the average segmentation precision was 94.646. The minimum results from the experimentation was a learning rate of .025, learning rate of decay of .01, learning rate decline of .7, batch size of 4, ROI of 64, 5 epochs, 1000 images per a class 10 classes and using the pipe dataset. The average precision of bounding boxes at these parameters was 71.564 and the average segmentation precision was 67.328. Table 14 is a table of the different generated maximizing optimization run results and one minimizing run result. These predictions are based on the average precision results for the 2^k factorial design. RoI, LRD, number of images per class, epochs, type and number of classes optimization results are similar, but the resulting factors differ.

Solutions									
Solution	LR	LRD	LRDecline	BATCH	ROI	IMPERCLASS	NUMCLASS	TYPE	EPOCH
1	0.025	0.10	.7	8	512	1500	5	1	15
2	0.025	0.01	.9	8	512	1500	5	1	15
3	0.025	0.01	.9	4	512	1500	5	1	15
4	0.001	0.01	.9	4	512	1500	5	1	15
5	0.001	0.01	.9	4	64	1500	5	1	15
6	0.025	0.01	.9	8	512	1000	5	1	15
7	0.001	0.01	.9	4	512	1000	5	1	15
8	0.025	0.01	.9	8	512	1500	5	1	5
9	0.025	0.10	.7	4	512	1500	5	1	15
10	0.025	0.01	.7	4	64	1000	10	5	1

Table 14: Optimization Results

Besides the optimization tool, Minitab also has a prediction tool to calculate the results of the regression model for imputed factors along with the confidence interval and prediction intervals. The bounding box model prediction for .025

learning rate, learning rate of decay of .1, learning rate decline of .7, batch size of 8, ROI of 512, 15 epochs, 1500 images per a class, 20-classes and for the Lego dataset bounding box prediction was 80.1017 % and the segmentation model prediction was 77.9926%. When a 20-class dataset was trained with solution 1 optimal training parameters, there was a CUDA out of memory error. The volume of data along with the number of images getting passed through the network at a time tried to allocate 392.00 MiB when there was only 84.62 MiB free of the 5.34 GiB reserved in total by PyTorch. The desktop itself has a 7.76 GiB capacity. When using the optimal settings from the factorial design on a 20 class Lego dataset, it predicted a segmentation result of 76.9422 with a 95% confidence interval of (76.485,77.4159) with the segmentation regression model. When the model was trained with a .025 learning rate, learning rate of decay of .1, learning rate decline of .7, batch size of 4, ROI of 512, 15 epochs, 1500 images per a class, the bounding box precision results predicted with the bounding box regression model the bounding box AP as 78.3157 with a 95% confidence interval (77.8143, 78.8170). This is slightly lower than the results from the preliminary analysis 11 which used a learning rate of .2, a batch size of 4 and ROI of 512. The main difference is that the epoch size is over 2 the size of the number of epochs used for this experiment. The number of images used was also a little over half of the number of images used for the experimental design.

When the experiment was run with solution 9 (maximum optimization) and solution 10 (minimum optimization) from Table 14, the actual segmentation and bounding box results were different from the prediction tool's results. When trained with solution 9 factors, the models result was 84.03 and segmentation result was 82.37. The original optimization prediction with equation 1 from Table 15 are a closer prediction to the actual result than a prediction using 4 batches

Equation ID	Model Type							
1	Two factorial regression model comparing test average precision vs. 9 Factors and Interactions							
2	Two factorial regression model comparing test average precision vs. 9 Factors							
3	Iteration Factors General Linear Model: Number of Classes, Images per Class, Batch and Epoch							
4	Hyperparameters General Linear Model: Learning Rate, Learning Rate Decline, Batch, ROI Epochs and Interactions							
5	Dataset Factors General linear model: Images per class, number of classes, type of dataset and interactions							
6	5 Class General linear models: LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions							
7	10 Class General linear models: LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions							
8	Lego General linear models: LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions height							
High Prediction					Low Prediction			
Equation ID	Bbox/AP		Difference bbox/AP	Difference Seg/AP	Bbox/AP		Difference bbox/AP	Difference Seg/AP
1	78.32	76.94	7%	7%	56.34	62.38	40%	25%
2	73.66	73.20	14%	13%	64.74	67.31	22%	15%
3*	84.64	83.23	-1%	-1%	82.00	80.80	-4%	-4%
4	101.25	94.86	-17%	-13%	73.92	77.20	7%	1%
5*	84.61	83.07	-1%	-1%	82.11	81.66	-4%	-5%
6	85.40	83.52	-2%	-1%	74.01	76.92	6%	1%
7	90.31	87.53	-7%	-6%	73.50	76.58	7%	2%
8*	88.22	85.91	-5%	-4%	73.74	77.82	7%	0%

*Predicted for 10 classes

Table 15: Prediction Results Comparison

instead of 8 results. The prediction tool result prediction for the maximum prediction (high) and minimum prediction (low) are in table 15. The real value is 7% higher than the predicted value for the high optimization variables but for the low optimization variables it is 40 - 25 % lower than the actual result for the Two factorial regression model comparing test average precision vs. 9 Factors and Interactions. A better prediction model to the actual result was equations 3, 5, 6. It was assumed that the iteration factors and dataset factors were important parameters and good indications of how the model will train. The 5-class model prediction being close to the actual result should be researched more to determine if the number of classes matters for the prediction model.

In order to verify that the 2^k factorial design regression models were significant for the data, multiple regression and general linear models were performed. Table 15 outlines the different types of equations used to predict the 20 class trial results for the high optimization parameters and the low optimization parameters. The difference from the actual result to the predicted result is presented next to the precision metrics. Only models whose normal probability plot were normal and residuals vs fits plots were randomly distributed are presented for analysis. The model from the factorial analysis does not predict the precision results. The

iteration factors model, the dataset models factor, the 5-class model and the Lego general linear model were closest to the actual result. This doesn't mean that they are better predictors - most of these models were run to compare results to determine if they influence how the object detection network works. Further explanation of the models is as follows.

Two factorial regression models were used to predict bounding box results and segmentation results by the 9 factors alone. The results of this analysis are in Appendix F. The model's R-squared values barely hover at 50% and error makes up half of the regression model. The regression models including factors interactions error value are low. The ANOVA analysis of the one-way factorial design for bounding box AP determined only learning rate, learning rate of decay, ROI, number of classes, type of dataset, and epochs as being significant. The ANOVA analysis of factorial design for segmentation AP determined only ROI, Number of classes, type of dataset, and epochs as being significant. The number of classes influenced both models the most. It wasn't expected that batch would be insignificant because the batch size was used to calculate the number of iterations used to train the models.

A separate analysis of the variables used to calculate the iterations for each experiment, Epochs, number of classes, number of images per class compared to the bounding box and segmentation responses. The models produced by this analysis can be seen in Appendix F. The models were not a strong fit for the data but were used to estimate the response for a 20-class dataset. The bounding box linear model's significant factors are number of classes, epochs, images per class x epoch, batch x number of classes x epoch and images per class x batch x epoch. For the segmentation model only number of classes, epoch, and images per class x epoch were significant.

General linear models separating factors that represent variation in the input data and from hyperparameters used to train the neural networks were also analyzed. The factors that represent variation in the input data produced models with slightly lower R-squared values of 90.94% for segmentation results and a 73.12% for bounding box results. A general linear model of the bounding box vs Learning Rate, Learning Rate Decline, Batch, ROI Epochs and Interactions produced a R-squared value of 99.77%. The factors found to be insignificant were LR x Learning rate decline, learning rate decline x ROI, and LRD x batch x ROI. A general linear model of the segmentation vs Learning Rate, Learning Rate Decline, Batch, ROI Epochs and Interactions produced a R-squared value of 99.76%. The factors found to be insignificant were learning rate, learning rate decay x epochs and batch x epochs. The Minitab results are in Appendix G.

A general linear model of the bounding box vs Images per class, number of classes, type of dataset and interactions found all factors significant besides the interaction between number of images per class and dataset type. This interaction was not picked up by other models because the interactions couldn't be estimated. The lack of significance is not surprising because the distinction between datasets shouldn't be size dependent. The general linear model of the segmentation vs Images per class, number of classes, type of dataset and interactions, however, was left out due the lack of normality of the linear model. The Minitab results of both analyses including residual plots are in Appendix G.

General linear models for segmentation and bounding box vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of dataset, epochs and interactions for 5 classes produced a viable model with R-squared values of about 99%. For the bounding box model, images per class was insignificant and for the segmentation model learning rate was insignificant. The bounding box result was not affected by

Figure 35: Two-Sample T-Test Box Plots

more input data but influenced segmentation results. The 10-class general linear models for segmentation and bounding box vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Dataset, Epochs and Interactions also produced viable models. For the segmentation model, learning rate, learning rate decline batch and LRD x ROI were insignificant. The Minitab results for all these models are found in Appendix G. A general linear model for each dataset type was also generated and is in Appendix G. The pipes models indicated that batch size was insignificant but in the Lego models Batch size is significant. Bounding box results for Legos indicated the number of classes was insignificant.

Figure 35 is the box plots of the average precision by number of classes as well by dataset type. The two-sample t-test that goes along with the box plots all reject the null hypothesis with p values less than .05, meaning they are not statistically the same. The full Minitab output is in Appendix H. In order to further verify that there is a statistical difference between the different trials beyond the regression and general linear analysis, a one-way ANOVA of the Lego and Pipe based trials was performed. The experimental means by bounding box results of each dataset are in Appendix I. The mean difference, t-value, degrees of freedom, and p-value

defined by the Games-Howell test are also in appendix I. The Games-Howell test does not assume equal variance when comparing the experimental groups. There were 32 experiments. Figure 36 is the pairwise comparison of the experiment’s groups by dataset type and the full set of experiments. The dataset types are broken down by iteration. The differentiation between experimental groups is prevalent. All the models’ Welch’s test had a p-value of zero and R-squared values above 90%.

Figure 36: Bounding Box Games-Howell Pairwise Comparisons

Lego’s models batch size significance could be due to the complexity of the Legos compared to the pipes or may not be a significant factor. Bounding box results for Legos indicated the number of classes oddly was insignificant. If this were accurate, the model results should be similar when run with the same number of classes.

An experiment was run to see if the average precision rises when the class number increases with the pipe dataset. Models trained for 1 class had a bounding box average precision of 6% higher than models trained for 2 classes. The 1 class model was 16% higher than a model trained for 3 classes. There was a 10% difference in models trained with 2 classes to 3 classes.

The 10-pipe dataset experimentation was performed with data labeling errors twice. The first was a dataset trained for 10 objects, but, was 8 different objects

and two replicate objects. The second was a dataset trained for 10 objects but was 9 different objects with 1 replicate object. The experiments were trained for 10 objects, but the multiple instances of an object caused them to be named as different class names. The results of the false experimentation as well as the correct experimentation are in Table 16.

10 Class Models :						
	8 different objects, 2 replicate objects		9 different objects, 1 replicate object		10 Different Object	
Iterations	bbox AP	Segmentation AP	bbox AP	Segmentation AP	bbox AP	Segmentation AP
6245	62.89122	65.63112	67.3712	70.1976	71.39287	74.99422
12497	69.06845	68.41499	74.8129	73.9966	80.61694	80.06845
18741	70.04771	69.19459	76.028	75.0159	82.15641	81.53204
37497	69.57589	69.81613	75.7446	75.9016	81.54909	82.01858

Table 16: Labeling Error Results

This table is not all the data from the project. It is the means of the trials used to compare the incorrectly formatted data. The precision for a dataset trained for 10 objects but with 9 different objects is 6-9% higher than a model trained with 8 different objects and two replicate objects for 10 classes. From the 10 objects with 9 different objects with 1 replicate to the model trained with 10 different objects there was a 6-9% increase in precision. This is unexpected. As the number of classes increased, the average precision typically decreased. In the cases of the false trials, there were still 10 classes but 8 objects. It would be expected that the fewer number of objects would result in a higher precision, but because of the mislabeling it was the opposite result. However, when the test data was relabeled to represent one object class for the duplicate object instances, the prediction metrics rose 21% for the two duplicate class dataset and 7% for the 1 replicate class dataset. Before the change, the model identified the identical object as the 1st or 2nd named class for the object and test data labeled specific instances of replicate objects as one class. The number of false positives caused by 2 classes having the same geometry had the same effect on the precision metrics as adding another class to the dataset. In other words, when a model is trained with 2 separate classes with the same 3D

Figure 37: Graphic of Labeling Error

CAD file, the average precision decreases by about 6-9% per duplicate object. When relabeled, the models increased to the typical 10 class precision results and even surpassed the 10 class 10 object databases in the 8-object case. Table 37 illustrates the relationship between the classes and replicate objects with simple shapes instead of pipe objects.

The label correction was calculated by running the test twice for each replicate object mistake. This was done to see if the weights per each class the same effect on the model had even though they were the same object. An example of this revision is the test JSON dataset file that was changed to replace instances of class 11 to 6, resulting in bounding box precision of 74.22 and segmentation precision of 77.22. If the model predicted the object to be class 11, it would be registered as a false positive, while it could be a true positive. Figure 17 shows the percent

Number of Objects	bbox_AP	Segm_AP
1	98.99	97.76
2	93.80	92.52
3	85.58	86.52

Table 17: Class Comparison

increase that occurred, but with only 3 objects. The prediction is a true positive

10 Class Models :						
	8 different objects, 2 replicate objects		9 different objects, 1 replicate object		10 Different Object	
Trial	bbox AP	Segmentation AP	bbox AP	Segmentation AP	bbox AP	Segmentation AP
Replicates	62.89122	65.63112	67.3712	70.1976	71.39287	74.99422
Relabeled Results	76.21993	79.20143	74.30726	77.33472		
Difference	+13.33%	+13.57%	+6.94%	+7.13%		

Table 18: Relabeled Results

only if it has a confidence score greater than the threshold, the predicted class matches the class of a ground truth and the predicted bounding box has an IoU greater than a threshold with the ground-truth [1]. When the reverse is replaced, the bounding box precision is 74.39 and the segmentation precision is 77.45. If the model predicted the object to be class 6, it would be registered as a false positive. The difference between the two types of changes was considered negligible because it was less than .3%; .175 for bounding box and .2289 for segmentation.

This change in results is illustrated in Table 18 with the means of the trials. The conclusion from this analysis is that the mislabeled replicates influence the pipe dataset in the same magnitude as adding a class. Adding a class or using a duplicate class lowers the average precision by 6-9%.

The difference from the 5-class pipe dataset to the 10-class pipe from the experimental design was 11-35% depending on the trial parameters. The average difference was 22%. That is an estimated 4.25% decrease per a class. The percent decrease is not precise but is an estimation. For the Lego dataset, the average precision increased and decreased in some cases but for the most part stayed the same. The average difference was -1% which is a .2% decrease per class.

The average precision may change by the factors presented but there is a time pay-off to increased precision. The scatter plots in Figure 38 depict the different experimentation as iterations. The time in hours it took to train the model is the x axis and the precision is the y-axis. An increased training time does not necessarily mean increased precision. The highest scoring metrics were with models run under

(a) bbox vs. Total training speed Scatter-plot

(b) Segmentation Results vs. Total Training Speed Scatterplot

Figure 38: Precision vs. Time

5 hours.

The factorial design data was analyzed for training speed vs experimentation. The results of this analysis are in Appendix D. The R-squared value is above 95%. All factors are analyzed besides Learning rate of decline. The Pareto Chart of the Standardized Effects of training speed on the model are in Figure 39. The most significant factor affecting training speed is batch size. The second most significant factor was ROI then batch x ROI. Batch's significance on the training speed length is reasonable because batch size is used to determine iterations. The larger the batch size, the less iterations necessary to run all the data through the network.

Figure 39: Training Speed Effects Graph

List of References

- [1] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: <http://arxiv.org/abs/1405.0312>

CHAPTER 6

Conclusion and Further Research

The Blenderproc synthetic object detection pipeline allows models to understand the 3D spatial information necessary for rich interactions with the physical world. Deep learning object detection models can be trained with synthetic data to recognize parts for manufacturing with an average precision greater than 80%. The controlled and customized object detection pipeline outperforms Detectron2 benchmark models COCO metrics. The results of the 2^k factorial design show the factors with the largest effect on the model are: Learning rate of decay times Learning Rate Decline, Number of Classes, type of dataset, epoch, Learning Rate times ROI, ROI. The ideal values for these factors depend on the mixture of factors but in most cases 1500 images per a class, a ROI of 512 and 15 epochs is enough to train a NN. With these parameters the learning rate produce similar precision result if set to .25 or .1. The learning rate decline point at 70% of the total number of images and set to .1. The batch size should be set to 4 or 8 depending on the number of classes being used to train a network. Generally larger datasets with many classes will have to use 4 images per batch. 40 depicts the visualized results of the 2K factorial design's optimal and lest optimal parameters as well as the generated images used for training. If a new model needs to be trained, the iterations used to train the network should be based off the optimization results. This is just a baseline solution, but due to the variability between datasets, further parameter optimization techniques may need to be performed if high prediction results are of value. However, these factors should result in a model accurate enough for implementation.

The object detection pipelines developed were an experimental baseline. The

future development to the pipeline is limitless and inevitable due to the rapid development of AI technology. Any improvement over 1% in the model is considered significant in the deep learning field. Advanced hardware enabling faster processing power enables phones and edge devices to run deep learning algorithms. The increased dependency on computers and telecommunications, especially during a global pandemic, drives digital transformation. During the COVID-19 pandemic, technology companies were significantly affected. The cloud services industry rapidly increased due to the increased use of technology and accelerated adoption of cloud, SaaS, and edge intelligence devices[1]. For manufacturing, the main concern is time and money and digitizing manufacturing. However, it enables companies to gain insights into production and operations management, which many argue is worth the investment. AI technologies can be implemented for direct automation, predictive maintenance, downtime reduction, 24/7 production, safety improvement, lower operational costs, improved efficiency, quality control, and accelerated decision making.

The tech industry connecting businesses with AI has been a critical factor to driving the decisions tech companies make to help recover and handle COVID-19. According to a 2019 Kentik report, 58 percent of businesses are already using a combination of Amazon Web Services (AWS), Microsoft Azure, and Google Cloud in their multi cloud networks [2]. Companies are focusing on cross-platform testing and allow technologies to operate across multiple environments. Enabling the interoperability of AI tools is the Open Neural Network Exchange (ONNX). ONNX was originally developed and open-sourced by Microsoft and Facebook in 2017 and now more than 30 companies contribute to the ONNX code base, including AWS, AMD, ARM, Baudi, HPE, IBM, Nvidia and Qualcomm. An ONNX conversion script should be added to the object detection pipeline. This will enable researchers

to be able to deploy more types of models, use quantify data formats, implement models with different networks, deploy models on mobile devices, and export to devices for optimized inference on specific devices [3]. Most of the pipeline is developed with open-sourced frameworks and models and trained locally. The process can easily be implemented as a cloud-based API but there is are growing concerns regarding cloud security. Data security was an important aspect to this project.

In the future, the pipeline could run on multiple computer systems with varying processing power and cloud computing services should be compared for precision and performance speeds. Evaluation scripts should be added to the pipeline to enable video recognition and edge deployment. Compression models, autoencoders, feature extractors and augmentation feature models can be worked into deep learning networks. The synthetic generation pipelines can be improved by adding realistic features and adding varying scenes per dataset. The dataset generation can be tested to further understand how training data affects the efficiency of the model. This can be done by varying object distance, camera position, pixel quality, etc. The different testing data parameters should also be compared. The performance of a model based on the dataset type and number of classes is of higher interest to the group. Further experimentation with larger datasets should help to understand the deep learning process. The 2K factorial DOE can be improved by adding center points to the experiment to ensure the linearity of the data. Further experimentation can focus on network development, large-scale synthetically trained models, models trained with 3D CAD data and depth information. Panoptic segmentation can also be experimented with.

Overall, pre-trained models can identify parts once trained with synthetically generated images. The deployment for object detection can be used for most

manufactured parts. AI is everywhere and its uses are constantly evolving. Society must keep up with the every-changing technological environment, and this project provides the steppingstones to do so.

Figure 40: 20 Lego Class Results

List of References

- [1] “Covid-19 outlook for the us technology industry,” Nov 2020. [Online]. Available: <https://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/covid-19-outlook-on-technology-industry.html>
- [2] “The state of automation, artificial intelligence and machine learning in network management,” 2020.
- [3] J. Spisak, 2018. [Online]. Available: <https://developers.facebook.com/blog/post/2018/05/02/onnx-expansion-speeds-research-production/>

APPENDIX A

Blender Pipeline Python Scripts

A.1 Create Automated Camera Rotation Script

```
import bpy

from math import pi, radians

bl_info = {
    "name": "Automated Camera Rotation",
    "category": "Object",
    "author": "Joel White/Jack Hurley",
    "version": (1, 0, 0),
    "blender": (2,80,0),
    "location": "Object > Create Automated Camera Rotation"
}

def createRotation(distance, horizontalSteps, verticalSteps, isHalfRotation):

    #create camera and CameraPath to rotate camera around
    bpy.ops.object.camera_add(location=(0,0,distance), align='VIEW',
rotation=(0,0,0))
    camera = bpy.context.object
    bpy.ops.object.empty_add(type='PLAIN_AXES', location=(0, 0, 0))
    CameraPath = bpy.context.object
    CameraPath.name = "Camera Rotator"
    camera.parent = CameraPath
    camera.matrix_parent_inverse = CameraPath.matrix_world.inverted()

    #Top frame
    bpy.context.scene.frame_set(0)
    CameraPath.rotation_euler = (0,0,0)
    CameraPath.keyframe_insert(data_path="rotation_euler")

    #loop for vertical rotation
    for i in range(verticalSteps - 1):

        #beginning keyframe
        bpy.context.scene.frame_set(bpy.context.scene.frame_current + 1)
    #set keyframe
        CameraPath.rotation_euler = (radians((180/verticalSteps) * (i + 1)),0,0)
    #rotate camera
        CameraPath.keyframe_insert(data_path="rotation_euler")
    #add keyframe

        #ending keyframe
        bpy.context.scene.frame_set(bpy.context.scene.frame_current +
horizontalSteps - 1)
        if(not isHalfRotation):
            CameraPath.rotation_euler = (radians((180/verticalSteps) * (i +
1)),0,radians(360 - (360/horizontalSteps)))
        else:
```

```

        CameraPath.rotation_euler = (radians((180/verticalSteps) * (i +
1)),0,radians(360 - (360/horizontalSteps))/2) #rotate camera
        CameraPath.keyframe_insert(data_path="rotation_euler")
                                #add keyframe

#Bottom frame
bpy.context.scene.frame_set(bpy.context.scene.frame_current + 1) #set
keyframe
    CameraPath.rotation_euler = (pi,0,0) #rotate
camera
    CameraPath.keyframe_insert(data_path="rotation_euler") #add
keyframe

#set frame length to total length
bpy.context.scene.frame_end = (bpy.context.scene.frame_current)
bpy.context.scene.frame_start = 0
bpy.context.scene.frame_set(0)

#make all animations linear
fcurses = CameraPath.animation_data.action.fcurves
for fcurve in fcurses:
    for kf in fcurve.keyframe_points:
        kf.interpolation = 'LINEAR'

class CameraRotation(bpy.types.Operator):
    """Create camera rotation animation""" #tooltip for menu items and
buttons.
    bl_idname = "object.automated_camera_rotation" # Unique identifier for buttons
and menu items to reference.
    bl_label = "Create Automated Camera Rotation" # Display name in the interface.
    bl_options = {'REGISTER', 'UNDO'} # Enable undo for the operator.

    distance = bpy.props.FloatProperty(name="Distance", default=75, soft_min=0)
    horizontalSteps = bpy.props.IntProperty(name="Horizontal Steps", default=180,
min=1)
    verticalSteps = bpy.props.IntProperty(name="Vertical Steps", default=36, min=1)
    isHalfRotation = bpy.props.BoolProperty(name="Use Half Rotation", default=False)

    def draw(self, context):
        layout = self.layout

        layout.prop(self, "distance")
        layout.separator()
        if(self.isHalfRotation):
            layout.label(text="Horizontal Angle: " +
str(round(180/self.horizontalSteps, 3)))
        else:

```



```

        layout.label(text="Horizontal Angle: " +
str(round(360/self.horizontalSteps, 3)))

        layout.prop(self, "horizontalSteps")
        layout.separator()
        layout.label(text="Vertical Angle: " + str(round(180/self.verticalSteps,3)))
        layout.prop(self, "verticalSteps")
        layout.separator()
        layout.prop(self, "isHalfRotation")

    def execute(self, context):          # execute() is called when running the
operator.

        # The original script

        createRotation(self.distance,self.horizontalSteps,self.verticalSteps,
self.isHalfRotation)

        return {'FINISHED'}            # Lets Blender know the operator finished
successfully.

# function to convert 2.7x code to work without warnings in 2.8x
def make_annotations(cls):
    """Converts class fields to annotations if running with Blender 2.8"""
    if bpy.app.version < (2, 80):
        return cls
    bl_props = {k: v for k, v in cls.__dict__.items() if isinstance(v, tuple)}
    if bl_props:
        if '__annotations__' not in cls.__dict__:
            setattr(cls, '__annotations__', {})
        annotations = cls.__dict__['__annotations__']
        for k, v in bl_props.items():
            annotations[k] = v
            delattr(cls, k)
    return cls

def menu_func(self, context):
    self.layout.operator(CameraRotation.bl_idname)

classes = (CameraRotation,)

def register():
    for cls in classes:
        make_annotations(cls)
        bpy.utils.register_class(cls)

```

```
bpy.types.VIEW3D_MT_object.append(menu_func)

def unregister():
    for cls in reversed(classes):
        bpy.utils.unregister_class(cls)

# This allows you to run the script directly from Blender's Text editor
# to test the add-on without having to install it.
if __name__ == "__main__":
    register()
```

A.2 Create Pascal VOC XML Files Script

```
import bpy
import math
import sys
import platform
import random
import os.path
from mathutils import Vector
import xml.etree.cElementTree as ET

bl_info = {
    "name": "Create Pascal VOC xml all images",
    "category": "Object",
    "author": "Joel White, Gabby Aiello",
    "version": (1, 0, 0),
    "blender": (2,80,0),
    "location": "Object>Create Pascal VOC xml all images"
}

def ShowMessageBox(message = "", title = "Message Box", icon = 'INFO'):

    def draw(self, context):
        self.layout.label(text=message)

    bpy.context.window_manager.popup_menu(draw, title = title, icon = icon)

#converts world coord to pixel coord
def World_To_Screen_Coord(
    world_point,
    camera_to_world,
    canvas_width,
    canvas_height,
    image_width,
    image_height):

    world_to_camera = camera_to_world.inverted()
    camera_point = world_to_camera @ world_point

    canvas_point = {}
    canvas_point[0] = camera_point.x / -camera_point.z
    canvas_point[1] = camera_point.y / -camera_point.z

    normalized_point = {}
    normalized_point[0] = (canvas_point[0] + canvas_width / 2) / canvas_width
    normalized_point[1] = (canvas_point[1] + canvas_height / 2) / canvas_height

    image_point = {}
    image_point[0] = math.floor(normalized_point[0] * image_width)
    image_point[1] = math.floor((1 - normalized_point[1]) * image_height)
```

```

return image_point

#creates xml file for each image in current animation timeline
def Create_XML_File(p_filepath, p_database, p_segmented, p_pose, p_truncated,
p_difficult, p_occluded, file_number, bndbox_list, test_numbers, file_extension,
obj_list, resolution_x, resolution_y, camera):

    is_test = False
    if int(file_number) in test_numbers:
        is_test = True

    if sys.platform.startswith('win'):
        image_name = p_filepath[p_filepath.rfind("\\")+1:] + file_number
        file_path = p_filepath[:p_filepath.rfind("\\")] + "\\VOC\\" + image_name
        folder_name = "VOC"
        """
        if is_test:
            image_name = p_filepath[p_filepath.rfind("\\")+1:] + file_number
            file_path = p_filepath[:p_filepath.rfind("\\")] + "\\testing\\" +
image_name
            folder_name = "testing"
        else:
            image_name = p_filepath[p_filepath.rfind("\\")+1:] + file_number
            file_path = p_filepath[:p_filepath.rfind("\\")] + "\\training\\" +
image_name
            folder_name = "training"
        """
    else:
        image_name = p_filepath[p_filepath.rfind("/") + 1:] + file_number
        file_path = p_filepath[:p_filepath.rfind("/")] + "/VOC/" + image_name
        folder_name = "VOC"
        """
        if is_test:
            image_name = p_filepath[p_filepath.rfind("/") + 1:] + file_number
            file_path = p_filepath[:p_filepath.rfind("/")] + "/testing/" +
image_name
            folder_name = "testing"
        else:
            image_name = p_filepath[p_filepath.rfind("/") + 1:] + file_number
            file_path = p_filepath[:p_filepath.rfind("/")] + "/training/" +
image_name
            folder_name = "training"
        """
    camera_angle = camera.matrix_world.to_euler('XYZ')

```

```

root = ET.Element("annotation")
folder = ET.SubElement(root, "folder").text = folder_name
filename = ET.SubElement(root, "filename").text = image_name + file_extension
path = ET.SubElement(root, "path").text = file_path + file_extension
source = ET.SubElement(root, "source")
database = ET.SubElement(source, "database").text = p_database

size = ET.SubElement(root, "size")
width = ET.SubElement(size, "width").text = str(resolution_x)[:2]
height = ET.SubElement(size, "height").text = str(resolution_y)[:2]
depth = ET.SubElement(size, "depth").text = "3"

segmented = ET.SubElement(root, "segmented").text = p_segmented
angle = ET.SubElement(root, "angle")
angle_x = ET.SubElement(angle, "xaxis").text = str(camera_angle[0])
angle_y = ET.SubElement(angle, "yaxis").text = str(camera_angle[1])
angle_z = ET.SubElement(angle, "zaxis").text = str(camera_angle[2])
"""
angle_x = ET.SubElement(angle, "xaxis").text = str(camera.global_coords[0])
angle_y = ET.SubElement(angle, "yaxis").text = str(camera.global_coords[1])
angle_z = ET.SubElement(angle, "zaxis").text = str(camera.global_coords[2])
"""

for i in range(len(obj_list)):
    object = ET.SubElement(root, "object")
    name = ET.SubElement(object, "name").text = obj_list[i].name
    pose = ET.SubElement(object, "pose").text = p_pose
    truncated = ET.SubElement(object, "truncated").text = p_truncated
    difficult = ET.SubElement(object, "difficult").text = p_difficult
    #occluded = ET.SubElement(object, "occluded").text = p_occluded
    bndbox = ET.SubElement(object, "bndbox")
    xmin = ET.SubElement(bndbox, "xmin").text = str(bndbox_list[i][0])
    ymin = ET.SubElement(bndbox, "ymin").text = str(bndbox_list[i][2])
    xmax = ET.SubElement(bndbox, "xmax").text = str(bndbox_list[i][1])
    ymax = ET.SubElement(bndbox, "ymax").text = str(bndbox_list[i][3])

tree = ET.ElementTree(root)
tree.write(file_path + ".xml")

print("Created " + file_path + ".xml")

def Move_Images(p_filepath, test_numbers, file_extension):
    for i in range(bpy.context.scene.frame_end - bpy.context.scene.frame_start + 1):
        number = ""
        if(i < 10000):
            number = f'{i:04}'

```

```

else:
    number = str(file_number)

if sys.platform.startswith('win'):
    if i in test_numbers:
        image_name = p_filepath[p_filepath.rfind("\\")+1:] + number
        file_path = p_filepath[:p_filepath.rfind("\\")] + "\\VOC\\" +
image_name
        os.rename(p_filepath + number + file_extension, file_path +
file_extension)
    else:
        image_name = p_filepath[p_filepath.rfind("\\")+1:] + number
        file_path = p_filepath[:p_filepath.rfind("\\")] + "\\VOC\\" +
image_name
        os.rename(p_filepath + number + file_extension, file_path +
file_extension)
    else:
        if i in test_numbers:
            image_name = p_filepath[p_filepath.rfind("/") + 1:] + number
            file_path = p_filepath[:p_filepath.rfind("/")] + "/VOC/" +
image_name
            os.rename(p_filepath + number + file_extension, file_path +
file_extension)
        else:
            image_name = p_filepath[p_filepath.rfind("/") + 1:] + number
            file_path = p_filepath[:p_filepath.rfind("/")] + "/VOC/" +
image_name
            os.rename(p_filepath + number + file_extension, file_path +
file_extension)

def Main_Function(p_filepath, p_database, p_segmented, p_pose, p_truncated,
p_difficult, p_occluded, p_training_percent):

    #set scene vars
    obj_list = bpy.context.selected_objects
    resolution_x = bpy.context.scene.render.resolution_x *
(bpy.context.scene.render.resolution_percentage/100)
    resolution_y = bpy.context.scene.render.resolution_y *
(bpy.context.scene.render.resolution_percentage/100)
    camera = bpy.context.scene.camera

    if(bpy.context.scene.render.image_settings.file_format == 'TIFF'):
        file_extension = ".tif"
    if(bpy.context.scene.render.image_settings.file_format == 'PNG'):
        file_extension = ".png"
    if(bpy.context.scene.render.image_settings.file_format == 'BMP'):
        file_extension = ".bmp"
    if(bpy.context.scene.render.image_settings.file_format == 'JPEG'):

```

```

        file_extension = ".jpg"
    if(bpy.context.scene.render.image_settings.file_format == 'TARGA'):
        file_extension = ".tga"
    if(bpy.context.scene.render.image_settings.file_format == 'OPEN_EXR'):
        file_extension = ".exr"

    #create folders
    # Create target Directory if don't exist
    if sys.platform.startswith('win'):
        filepath = p_filepath #full path, no extension
        image_name = filepath[filepath.rfind("\\")+1:] #name
        folder_name = filepath[:-len(image_name) - 1] #remove image
name

        if not os.path.exists(folder_name + "\\VOC"):
            os.mkdir(folder_name + "\\VOC")
            print("Directory ", folder_name + "\\VOC", " Created ")
        else:
            print("Directory ", folder_name + "\\VOC", " already exists")

    else:
        filepath = p_filepath #full path, no extension
        image_name = filepath[filepath.rfind("/") + 1:] #name
        folder_name = filepath[:-len(image_name) - 1] #remove image
name

        if not os.path.exists(folder_name + "/VOC"):
            os.mkdir(folder_name + "/VOC")
            print("Directory ", folder_name + "/VOC", " Created ")
        else:
            print("Directory ", folder_name + "/VOC", " already exists")

    #select random files for training/testing
    test_numbers = []

    for i in range(int((bpy.context.scene.frame_end - bpy.context.scene.frame_start
+ 1) * ((100 - p_training_percent)/100))):
        num = -1
        while True:
            num = random.randrange(bpy.context.scene.frame_end -
bpy.context.scene.frame_start + 1)
            if not (num in test_numbers):
                break

        test_numbers.append(num)

```

```

c_width = 2 * math.tan(camera.data.angle/2)
c_height = c_width * (resolution_y/resolution_x)

#convert local coords to world coords
world_vertices = []
for i in range(len(obj_list)):

    obj_vertices = []
    for k in range(len(obj_list[i].data.vertices)):
        obj_vertices.append(obj_list[i].matrix_world @
obj_list[i].data.vertices[k].co)

    world_vertices.append(obj_vertices)

#create xml files for each frame
for i in range(bpy.context.scene.frame_end - bpy.context.scene.frame_start + 1):

    bpy.context.scene.frame_set(i)

    bndbox_list = [] #list of x/y min/max

    for k in range(len(obj_list)):

        #initilize vars
        x_min = sys.maxsize
        x_max = -sys.maxsize - 1
        y_min = sys.maxsize
        y_max = -sys.maxsize - 1

        #find min/max x/y
        min_max_list = []
        for x in range(len(world_vertices[k])):

            pos = World_To_Screen_Coord(world_vertices[k][x],
camera.matrix_world, c_width, c_height, resolution_x, resolution_y)

            if(pos[0] < x_min):
                x_min = pos[0]
            if(pos[0] > x_max):
                x_max = pos[0]
            if(pos[1] < y_min):
                y_min = pos[1]
            if(pos[1] > y_max):
                y_max = pos[1]

        min_max_list = [x_min,x_max,y_min,y_max]

```



```

bndbox_list.append(min_max_list)

file_number = i + bpy.context.scene.frame_start;
if(file_number < 10000):
    file_number = f'{file_number:04}'
else:
    file_number = str(file_number)

Create_XML_File(p_filepath, p_database, p_segmented, p_pose, p_truncated,
p_difficult, p_occluded, file_number, bndbox_list, test_numbers, file_extension,
obj_list, resolution_x, resolution_y, camera);

try:
    Move_Images(p_filepath, test_numbers, file_extension)
except:
    ShowMessageBox("Unable to move images into 'training'/'testing' folders",
"Pascal VOC xml Warning", 'ERROR')

class VOCxml(bpy.types.Operator):
    """Create Pascal VOC xml Files all images"""
    bl_idname = "object.pascal_voc_xml_all_images"
    bl_label = "Create Pascal VOC xml Files all images"
    bl_options = {'REGISTER', 'UNDO'}

    training_percentage = bpy.props.FloatProperty(name="Training Percentage",
subtpe="PERCENTAGE", precision=1, min=0, max=100, default=100)
    filepath = bpy.props.StringProperty(name="File Path", subtype="FILE_PATH",
options={"HIDDEN"})
    database = bpy.props.StringProperty(name="Database", default="Unspecified")
    segmented = bpy.props.StringProperty(name="Segmented", default="0")
    pose = bpy.props.StringProperty(name="Pose", default="Unspecified")
    truncated = bpy.props.StringProperty(name="Truncated", default="0")
    difficult = bpy.props.StringProperty(name="Difficult", default="0")
    occluded = bpy.props.StringProperty(name="Occluded", default="0")

    @classmethod
    def poll(cls, context):
        return context.object.select_get() and context.object.type == 'MESH'

    def invoke(self, context, event):
        context.window_manager.fileselect_add(self)
        #return context.window_manager.invoke_props_dialog(self)
        return {'RUNNING_MODAL'}

```

```

def execute(self, context):
    print("PATH: " + self.filepath)
    print("DATABASE: " + self.database)
    print("SEGMENTED: " + self.segmented)
    print("POSE: " + self.pose)
    print("TRUNCATED: " + self.truncated)
    print("DIFFICULT: " + self.difficult)
    print("OCCLUDED: " + self.occluded)
    Main_Function(self.filepath, self.database, self.segmented, self.pose,
self.truncated, self.difficult, self.occluded, 100)
    return {'FINISHED'}

# function to convert 2.7x code to work without warnings in 2.8x
def make_annotations(cls):
    """Converts class fields to annotations if running with Blender 2.8"""
    if bpy.app.version < (2, 80):
        return cls
    bl_props = {k: v for k, v in cls.__dict__.items() if isinstance(v, tuple)}
    if bl_props:
        if '__annotations__' not in cls.__dict__:
            setattr(cls, '__annotations__', {})
        annotations = cls.__dict__['__annotations__']
        for k, v in bl_props.items():
            annotations[k] = v
            delattr(cls, k)
    return cls

def menu_func(self, context):
    self.layout.operator(VOCxml.bl_idname)

classes = (VOCxml,)

def register():
    for cls in classes:
        make_annotations(cls)
        bpy.utils.register_class(cls)

    bpy.types.VIEW3D_MT_object.append(menu_func)

def unregister():
    for cls in reversed(classes):
        bpy.utils.unregister_class(cls)

```

```
if __name__ == "__main__":  
    register()
```

APPENDIX B

TensorFlow Fixtures Object Detection Jupyter Notebook Output

1 Object Detection Demo

Welcome to the object detection inference walkthrough! This notebook will walk you step by step through the process of using a pre-trained model to detect objects in an image. Make sure to follow the [installation instructions](#) before you start.

2 Imports

```
[1]: %%time
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from distutils.version import StrictVersion
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

plt.rcParams.update({'figure.max_open_warning': 0})
# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.12.0'):
    raise ImportError('Please upgrade your TensorFlow installation to v1.12.*.')
```

Wall time: 2.18 s

2.1 Env setup

```
[2]: # This is needed to display the images.
%matplotlib inline
```

2.2 Object detection imports

Here are the imports from the object detection module.

```
[3]: from utils import label_map_util

from utils import visualization_utils as vis_util
```

3 Model preparation

3.1 Variables

Any model exported using the `export_inference_graph.py` tool can be loaded here simply by changing `PATH_TO_FROZEN_GRAPH` to point to a new `.pb` file.

By default we use an “SSD with Mobilenet” model here. See the [detection model zoo](#) for a list of other models that can be run out-of-the-box with varying speeds and accuracies.

```
[4]: %%time

# What model to download.
MODEL_NAME = 'inference_graph'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for the
# object detection.
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'object-detection.pbtxt')
```

Wall time: 0 ns

3.2 Download Model

3.3 Load a (frozen) Tensorflow model into memory.

```
[5]: %%time

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
```

Wall time: 2.92 s

3.4 Loading label map

Label maps map indices to category names, so that when our convolution network predicts 5, we know that this corresponds to airplane. Here we use internal utility functions, but anything that returns a dictionary mapping integers to appropriate string labels would be fine

```
[6]: category_index = label_map_util.  
      ↪create_category_index_from_labelmap(PATH_TO_LABELS, use_display_name=True)
```

WARNING: Logging before flag parsing goes to stderr.
W1029 12:10:53.532942 44988 deprecation_wrapper.py:119] From
D:\tensorflow1\fixture1\utils\label_map_util.py:137: The name tf.gfile.GFile is
deprecated. Please use tf.io.gfile.GFile instead.

3.5 Helper code

```
[7]: def load_image_into_numpy_array(image):  
      (im_width, im_height) = image.size  
      ↪return np.array(image.getdata()).reshape(  
          (im_height, im_width, 3)).astype(np.uint8)
```

4 Detection

```
[8]: %%time  
  
      # For the sake of simplicity we will use only 2 images:  
      # image1.jpg  
      # image2.jpg  
      # If you want to test the code with your images, just add path to the images to ↪  
      ↪the TEST_IMAGE_PATHS.  
      PATH_TO_TEST_IMAGES_DIR = 'test_images'  
      TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, each_image) for ↪  
          ↪each_image in os.listdir("D:/tensorflow1/fixture1/test_images") ]  
      print(TEST_IMAGE_PATHS)  
      print(len(TEST_IMAGE_PATHS))  
      # Size, in inches, of the output images.  
      IMAGE_SIZE = (12, 8)  
  
      ['test_images\\IMG_2294.JPG', 'test_images\\IMG_8477.JPG',  
      'test_images\\IMG_8478.JPG', 'test_images\\IMG_8479.JPG',  
      'test_images\\IMG_8508.JPG', 'test_images\\IMG_8509.JPG',  
      'test_images\\IMG_8510.JPG', 'test_images\\IMG_8511.JPG',  
      'test_images\\IMG_8512.JPG', 'test_images\\IMG_8513.JPG']  
      10  
      Wall time: 2 ms
```

```
[9]: %%time  
  
      def run_inference_for_single_image(image, graph):  
          with graph.as_default():  
              with tf.Session() as sess:
```

```

# Get handles to input and output tensors
ops = tf.get_default_graph().get_operations()
all_tensor_names = {output.name for op in ops for output in op.outputs}
tensor_dict = {}
for key in [
    'num_detections', 'detection_boxes', 'detection_scores',
    'detection_classes', 'detection_masks'
]:
    tensor_name = key + ':0'
    if tensor_name in all_tensor_names:
        tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
            tensor_name)
    if 'detection_masks' in tensor_dict:
        # The following processing is only for single image
        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
        # Reframe is required to translate mask from box coordinates to image
        ↪coordinates and fit the image size.
        real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
        detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection,
        ↪-1])
        detection_masks = tf.slice(detection_masks, [0, 0, 0],
        ↪[real_num_detection, -1, -1])
        detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
            detection_masks, detection_boxes, image.shape[1], image.shape[2])
        detection_masks_reframed = tf.cast(
            tf.greater(detection_masks_reframed, 0.5), tf.uint8)
        # Follow the convention by adding back the batch dimension
        tensor_dict['detection_masks'] = tf.expand_dims(
            detection_masks_reframed, 0)
    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')

# Run inference
output_dict = sess.run(tensor_dict,
                        feed_dict={image_tensor: image})

# all outputs are float32 numpy arrays, so convert types as appropriate
output_dict['num_detections'] = int(output_dict['num_detections'][0])
output_dict['detection_classes'] = output_dict[
    'detection_classes'][0].astype(np.int64)
output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
output_dict['detection_scores'] = output_dict['detection_scores'][0]
if 'detection_masks' in output_dict:
    output_dict['detection_masks'] = output_dict['detection_masks'][0]
return output_dict

```

Wall time: 0 ns

```
[ ]: %%time

for image_path in TEST_IMAGE_PATHS:
    image = Image.open(image_path)
    # the array based representation of the image will be used later in order to
    # prepare the
    # result image with boxes and labels on it.
    image_np = load_image_into_numpy_array(image)
    # Expand dimensions since the model expects images to have shape: [1, None,
    # None, 3]
    image_np_expanded = np.expand_dims(image_np, axis=0)
    # Actual detection.
    output_dict = run_inference_for_single_image(image_np_expanded,
    # detection_graph)
    # Visualization of the results of a detection.
    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        output_dict['detection_boxes'],
        output_dict['detection_classes'],
        output_dict['detection_scores'],
        category_index,
        instance_masks=output_dict.get('detection_masks'),
        use_normalized_coordinates=True,
        line_thickness=8)
    plt.figure(figsize=IMAGE_SIZE)
    plt.imshow(image_np)
    plt.show()
    print("D:/tensorflow1/fixture1/results/" + image_path[-8:-4] + "_3.JPG")
    plt.imsave("D:/tensorflow1/fixture1/results/" + image_path[-8:-4] + "_3.JPG",
    image_np)
```

C:\Users\URI_PC\Anaconda3\envs\object_det2\lib\site-packages\ipykernel_launcher.py:22: UserWarning: Matplotlib is currently using agg, which is a non-GUI backend, so cannot show the figure.

D:/tensorflow1/fixture1/results/2294_3.JPG
D:/tensorflow1/fixture1/results/8477_3.JPG
D:/tensorflow1/fixture1/results/8478_3.JPG
D:/tensorflow1/fixture1/results/8479_3.JPG
D:/tensorflow1/fixture1/results/8508_3.JPG
D:/tensorflow1/fixture1/results/8509_3.JPG
D:/tensorflow1/fixture1/results/8510_3.JPG
D:/tensorflow1/fixture1/results/8511_3.JPG
D:/tensorflow1/fixture1/results/8512_3.JPG
D:/tensorflow1/fixture1/results/8513_3.JPG

[]:

APPENDIX C

High and Low Variance Minitab P-Test Results

WORKSHEET 3

Paired T-Test and CI: hbb, lbb

Descriptive Statistics

Sample	N	Mean	StDev	SE Mean
hbb	12	0.8001	0.2413	0.0696
lbb	12	0.7781	0.2806	0.0810

Estimation for Paired Difference

Mean	StDev	SE Mean	95% CI for $\mu_{\text{difference}}$
0.0220	0.0401	0.0116	(-0.0035, 0.0475)

$\mu_{\text{difference}}$: mean of (hbb - lbb)

Test

Null hypothesis	$H_0: \mu_{\text{difference}} = 0$
Alternative hypothesis	$H_1: \mu_{\text{difference}} \neq 0$
T-Value	1.90
P-Value	0.084

Paired T-Test and CI: hseg1, lseg1

Descriptive Statistics

Sample	N	Mean	StDev	SE Mean
hseg1	12	0.7738	0.2455	0.0709
lseg1	12	0.7697	0.2863	0.0827

Estimation for Paired Difference

Mean	StDev	SE Mean	95% CI for $\mu_{\text{difference}}$
0.0041	0.0572	0.0165	(-0.0323, 0.0404)

$\mu_{\text{difference}}$: mean of (hseg1 - lseg1)

Test

Null hypothesis $H_0: \mu_{\text{difference}} = 0$
 Alternative hypothesis $H_1: \mu_{\text{difference}} \neq 0$

T-Value	P-Value
0.25	0.809

APPENDIX D

Test Dataset Size Comparison of Means Results

One-Way ANOVA Bounding box AP vs. Number of Images

WORKSHEET 4

One-way ANOVA: bb_AP versus image

Method

Null hypothesis All means are equal
Alternative hypothesis Not all means are equal
Significance level $\alpha = 0.05$

Equal variances were assumed for the analysis.

Factor Information

Factor	Levels	Values
image	3	250, 500, 700

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
image	2	0.9535	0.4768	0.76	0.509
Error	6	3.7721	0.6287		
Total	8	4.7256			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.792894	20.18%	0.00%	0.00%

Means

image	N	Mean	StDev	95% CI
250	2	81.063	0.529	(79.691, 82.435)
500	4	81.211	0.870	(80.241, 82.181)
700	3	80.482	0.781	(79.362, 81.602)

Pooled StDev = 0.792894

One-Way ANOVA Segmentation AP vs. Number of Images

WORKSHEET 4

One-way ANOVA: Segm_AP versus image

Method

Null hypothesis All means are equal
Alternative hypothesis Not all means are equal
Significance level $\alpha = 0.05$

Equal variances were assumed for the analysis.

Factor Information

Factor	Levels	Values
image	3	250, 500, 700

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
image	2	0.8836	0.4418	0.71	0.531
Error	6	3.7577	0.6263		
Total	8	4.6413			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.791383	19.04%	0.00%	0.00%

Means

image	N	Mean	StDev	95% CI
250	2	80.509	0.415	(79.139, 81.878)
500	4	80.309	0.923	(79.341, 81.278)
700	3	79.731	0.717	(78.613, 80.849)

Pooled StDev = 0.791383

APPENDIX E

2^k Factorial Design Results

frDB	RunOrder	CenterPt	Blocks	LR	LRD	LRDecline	Batch	ROI	Images per class	Num Class es	Type Dataset	Epochs	# total Images	# of Iterations
16	16	1	1	0.025	0.1	0.9	8	64	1000	5	Lego	5	5000	3120
48	48	1	1	0.025	0.1	0.9	8	64	1000	5	Lego	5	5000	3120
80	80	1	1	0.025	0.1	0.9	8	64	1000	5	Lego	5	5000	3120
112	112	1	1	0.025	0.1	0.9	8	64	1000	5	Lego	5	5000	3120
144	144	1	1	0.025	0.1	0.9	8	64	1000	5	Lego	5	5000	3120
31	31	1	1	0.001	0.1	0.9	8	512	1500	5	Lego	5	7500	4680
63	63	1	1	0.001	0.1	0.9	8	512	1500	5	Lego	5	7500	4680
95	95	1	1	0.001	0.1	0.9	8	512	1500	5	Lego	5	7500	4680
127	127	1	1	0.001	0.1	0.9	8	512	1500	5	Lego	5	7500	4680
159	159	1	1	0.001	0.1	0.9	8	512	1500	5	Lego	5	7500	4680
17	17	1	1	0.001	0.01	0.7	4	512	1000	5	Lego	5	5000	6248
49	49	1	1	0.001	0.01	0.7	4	512	1000	5	Lego	5	5000	6248
81	81	1	1	0.001	0.01	0.7	4	512	1000	5	Lego	5	5000	6248
113	113	1	1	0.001	0.01	0.7	4	512	1000	5	Lego	5	5000	6248
145	145	1	1	0.001	0.01	0.7	4	512	1000	5	Lego	5	5000	6248
9	9	1	1	0.001	0.01	0.7	8	64	1000	5	Lego	15	5000	9368
41	41	1	1	0.001	0.01	0.7	8	64	1000	5	Lego	15	5000	9368
73	73	1	1	0.001	0.01	0.7	8	64	1000	5	Lego	15	5000	9368
105	105	1	1	0.001	0.01	0.7	8	64	1000	5	Lego	15	5000	9368
137	137	1	1	0.001	0.01	0.7	8	64	1000	5	Lego	15	5000	9368
2	2	1	1	0.025	0.01	0.7	4	64	1500	5	Lego	5	7500	9372
34	34	1	1	0.025	0.01	0.7	4	64	1500	5	Lego	5	7500	9372
66	66	1	1	0.025	0.01	0.7	4	64	1500	5	Lego	5	7500	9372
98	98	1	1	0.025	0.01	0.7	4	64	1500	5	Lego	5	7500	9372
130	130	1	1	0.025	0.01	0.7	4	64	1500	5	Lego	5	7500	9372
26	26	1	1	0.025	0.01	0.7	8	512	1500	5	Lego	15	7500	14056
58	58	1	1	0.025	0.01	0.7	8	512	1500	5	Lego	15	7500	14056
90	90	1	1	0.025	0.01	0.7	8	512	1500	5	Lego	15	7500	14056
122	122	1	1	0.025	0.01	0.7	8	512	1500	5	Lego	15	7500	14056
154	154	1	1	0.025	0.01	0.7	8	512	1500	5	Lego	15	7500	14056
24	24	1	1	0.025	0.1	0.9	4	512	1000	5	Lego	15	5000	18748
56	56	1	1	0.025	0.1	0.9	4	512	1000	5	Lego	15	5000	18748
88	88	1	1	0.025	0.1	0.9	4	512	1000	5	Lego	15	5000	18748
120	120	1	1	0.025	0.1	0.9	4	512	1000	5	Lego	15	5000	18748
152	152	1	1	0.025	0.1	0.9	4	512	1000	5	Lego	15	5000	18748
7	7	1	1	0.001	0.1	0.9	4	64	1500	5	Lego	15	7500	28124
39	39	1	1	0.001	0.1	0.9	4	64	1500	5	Lego	15	7500	28124
71	71	1	1	0.001	0.1	0.9	4	64	1500	5	Lego	15	7500	28124
103	103	1	1	0.001	0.1	0.9	4	64	1500	5	Lego	15	7500	28124
135	135	1	1	0.001	0.1	0.9	4	64	1500	5	Lego	15	7500	28124
30	30	1	1	0.025	0.01	0.9	8	512	1000	10	Lego	5	10000	6248
62	62	1	1	0.025	0.01	0.9	8	512	1000	10	Lego	5	10000	6248
94	94	1	1	0.025	0.01	0.9	8	512	1000	10	Lego	5	10000	6248
126	126	1	1	0.025	0.01	0.9	8	512	1000	10	Lego	5	10000	6248

158	158	1	1	0.025	0.01	0.9	8	512	1000	10 Lego	5	10000	6248
13	13	1	1	0.001	0.01	0.9	8	64	1500	10 Lego	5	15000	9368
45	45	1	1	0.001	0.01	0.9	8	64	1500	10 Lego	5	15000	9368
77	77	1	1	0.001	0.01	0.9	8	64	1500	10 Lego	5	15000	9368
109	109	1	1	0.001	0.01	0.9	8	64	1500	10 Lego	5	15000	9368
141	141	1	1	0.001	0.01	0.9	8	64	1500	10 Lego	5	15000	9368
3	3	1	1	0.001	0.1	0.7	4	64	1000	10 Lego	5	10000	12500
35	35	1	1	0.001	0.1	0.7	4	64	1000	10 Lego	5	10000	12500
67	67	1	1	0.001	0.1	0.7	4	64	1000	10 Lego	5	10000	12500
99	99	1	1	0.001	0.1	0.7	4	64	1000	10 Lego	5	10000	12500
131	131	1	1	0.001	0.1	0.7	4	64	1000	10 Lego	5	10000	12500
27	27	1	1	0.001	0.1	0.7	8	512	1000	10 Lego	15	10000	18744
59	59	1	1	0.001	0.1	0.7	8	512	1000	10 Lego	15	10000	18744
91	91	1	1	0.001	0.1	0.7	8	512	1000	10 Lego	15	10000	18744
123	123	1	1	0.001	0.1	0.7	8	512	1000	10 Lego	15	10000	18744
155	155	1	1	0.001	0.1	0.7	8	512	1000	10 Lego	15	10000	18744
20	20	1	1	0.025	0.1	0.7	4	512	1500	10 Lego	5	15000	18748
52	52	1	1	0.025	0.1	0.7	4	512	1500	10 Lego	5	15000	18748
84	84	1	1	0.025	0.1	0.7	4	512	1500	10 Lego	5	15000	18748
116	116	1	1	0.025	0.1	0.7	4	512	1500	10 Lego	5	15000	18748
148	148	1	1	0.025	0.1	0.7	4	512	1500	10 Lego	5	15000	18748
12	12	1	1	0.025	0.1	0.7	8	64	1500	10 Lego	15	15000	28120
44	44	1	1	0.025	0.1	0.7	8	64	1500	10 Lego	15	15000	28120
76	76	1	1	0.025	0.1	0.7	8	64	1500	10 Lego	15	15000	28120
108	108	1	1	0.025	0.1	0.7	8	64	1500	10 Lego	15	15000	28120
140	140	1	1	0.025	0.1	0.7	8	64	1500	10 Lego	15	15000	28120
6	6	1	1	0.025	0.01	0.9	4	64	1000	10 Lego	15	10000	37500
38	38	1	1	0.025	0.01	0.9	4	64	1000	10 Lego	15	10000	37500
70	70	1	1	0.025	0.01	0.9	4	64	1000	10 Lego	15	10000	37500
102	102	1	1	0.025	0.01	0.9	4	64	1000	10 Lego	15	10000	37500
134	134	1	1	0.025	0.01	0.9	4	64	1000	10 Lego	15	10000	37500
21	21	1	1	0.001	0.01	0.9	4	512	1500	10 Lego	15	15000	56248
53	53	1	1	0.001	0.01	0.9	4	512	1500	10 Lego	15	15000	56248
85	85	1	1	0.001	0.01	0.9	4	512	1500	10 Lego	15	15000	56248
117	117	1	1	0.001	0.01	0.9	4	512	1500	10 Lego	15	15000	56248
149	149	1	1	0.001	0.01	0.9	4	512	1500	10 Lego	15	15000	56248
28	28	1	1	0.025	0.1	0.7	8	512	1000	5 Machine	5	5000	3120
60	60	1	1	0.025	0.1	0.7	8	512	1000	5 Machine	5	5000	3120
92	92	1	1	0.025	0.1	0.7	8	512	1000	5 Machine	5	5000	3120
124	124	1	1	0.025	0.1	0.7	8	512	1000	5 Machine	5	5000	3120
156	156	1	1	0.025	0.1	0.7	8	512	1000	5 Machine	5	5000	3120
11	11	1	1	0.001	0.1	0.7	8	64	1500	5 Machine	5	7500	4680
43	43	1	1	0.001	0.1	0.7	8	64	1500	5 Machine	5	7500	4680
75	75	1	1	0.001	0.1	0.7	8	64	1500	5 Machine	5	7500	4680
107	107	1	1	0.001	0.1	0.7	8	64	1500	5 Machine	5	7500	4680
139	139	1	1	0.001	0.1	0.7	8	64	1500	5 Machine	5	7500	4680
5	5	1	1	0.001	0.01	0.9	4	64	1000	5 Machine	5	5000	6248

37	37	1	1	0.001	0.01	0.9	4	64	1000	5 Machine	5	5000	6248
69	69	1	1	0.001	0.01	0.9	4	64	1000	5 Machine	5	5000	6248
101	101	1	1	0.001	0.01	0.9	4	64	1000	5 Machine	5	5000	6248
133	133	1	1	0.001	0.01	0.9	4	64	1000	5 Machine	5	5000	6248
29	29	1	1	0.001	0.01	0.9	8	512	1000	5 Machine	15	5000	9368
61	61	1	1	0.001	0.01	0.9	8	512	1000	5 Machine	15	5000	9368
93	93	1	1	0.001	0.01	0.9	8	512	1000	5 Machine	15	5000	9368
125	125	1	1	0.001	0.01	0.9	8	512	1000	5 Machine	15	5000	9368
157	157	1	1	0.001	0.01	0.9	8	512	1000	5 Machine	15	5000	9368
22	22	1	1	0.025	0.01	0.9	4	512	1500	5 Machine	5	7500	9372
54	54	1	1	0.025	0.01	0.9	4	512	1500	5 Machine	5	7500	9372
86	86	1	1	0.025	0.01	0.9	4	512	1500	5 Machine	5	7500	9372
118	118	1	1	0.025	0.01	0.9	4	512	1500	5 Machine	5	7500	9372
150	150	1	1	0.025	0.01	0.9	4	512	1500	5 Machine	5	7500	9372
14	14	1	1	0.025	0.01	0.9	8	64	1500	5 Machine	15	7500	14056
46	46	1	1	0.025	0.01	0.9	8	64	1500	5 Machine	15	7500	14056
78	78	1	1	0.025	0.01	0.9	8	64	1500	5 Machine	15	7500	14056
110	110	1	1	0.025	0.01	0.9	8	64	1500	5 Machine	15	7500	14056
142	142	1	1	0.025	0.01	0.9	8	64	1500	5 Machine	15	7500	14056
4	4	1	1	0.025	0.1	0.7	4	64	1000	5 Machine	15	5000	18748
36	36	1	1	0.025	0.1	0.7	4	64	1000	5 Machine	15	5000	18748
68	68	1	1	0.025	0.1	0.7	4	64	1000	5 Machine	15	5000	18748
100	100	1	1	0.025	0.1	0.7	4	64	1000	5 Machine	15	5000	18748
132	132	1	1	0.025	0.1	0.7	4	64	1000	5 Machine	15	5000	18748
19	19	1	1	0.001	0.1	0.7	4	512	1500	5 Machine	15	7500	28124
51	51	1	1	0.001	0.1	0.7	4	512	1500	5 Machine	15	7500	28124
83	83	1	1	0.001	0.1	0.7	4	512	1500	5 Machine	15	7500	28124
115	115	1	1	0.001	0.1	0.7	4	512	1500	5 Machine	15	7500	28124
147	147	1	1	0.001	0.1	0.7	4	512	1500	5 Machine	15	7500	28124
10	10	1	1	0.025	0.01	0.7	8	64	1000	10 Machine	5	10000	6248
42	42	1	1	0.025	0.01	0.7	8	64	1000	10 Machine	5	10000	6248
74	74	1	1	0.025	0.01	0.7	8	64	1000	10 Machine	5	10000	6248
106	106	1	1	0.025	0.01	0.7	8	64	1000	10 Machine	5	10000	6248
138	138	1	1	0.025	0.01	0.7	8	64	1000	10 Machine	5	10000	6248
25	25	1	1	0.001	0.01	0.7	8	512	1500	10 Machine	5	15000	9368
57	57	1	1	0.001	0.01	0.7	8	512	1500	10 Machine	5	15000	9368
89	89	1	1	0.001	0.01	0.7	8	512	1500	10 Machine	5	15000	9368
121	121	1	1	0.001	0.01	0.7	8	512	1500	10 Machine	5	15000	9368
153	153	1	1	0.001	0.01	0.7	8	512	1500	10 Machine	5	15000	9368
23	23	1	1	0.001	0.1	0.9	4	512	1000	10 Machine	5	10000	12500
55	55	1	1	0.001	0.1	0.9	4	512	1000	10 Machine	5	10000	12500
87	87	1	1	0.001	0.1	0.9	4	512	1000	10 Machine	5	10000	12500
119	119	1	1	0.001	0.1	0.9	4	512	1000	10 Machine	5	10000	12500
151	151	1	1	0.001	0.1	0.9	4	512	1000	10 Machine	5	10000	12500
15	15	1	1	0.001	0.1	0.9	8	64	1000	10 Machine	15	10000	18744
47	47	1	1	0.001	0.1	0.9	8	64	1000	10 Machine	15	10000	18744
79	79	1	1	0.001	0.1	0.9	8	64	1000	10 Machine	15	10000	18744

111	111	1	1	0.001	0.1	0.9	8	64	1000	10 Machine	15	10000	18744
143	143	1	1	0.001	0.1	0.9	8	64	1000	10 Machine	15	10000	18744
8	8	1	1	0.025	0.1	0.9	4	64	1500	10 Machine	5	15000	18748
40	40	1	1	0.025	0.1	0.9	4	64	1500	10 Machine	5	15000	18748
72	72	1	1	0.025	0.1	0.9	4	64	1500	10 Machine	5	15000	18748
104	104	1	1	0.025	0.1	0.9	4	64	1500	10 Machine	5	15000	18748
136	136	1	1	0.025	0.1	0.9	4	64	1500	10 Machine	5	15000	18748
32	32	1	1	0.025	0.1	0.9	8	512	1500	10 Machine	15	15000	28120
64	64	1	1	0.025	0.1	0.9	8	512	1500	10 Machine	15	15000	28120
96	96	1	1	0.025	0.1	0.9	8	512	1500	10 Machine	15	15000	28120
128	128	1	1	0.025	0.1	0.9	8	512	1500	10 Machine	15	15000	28120
160	160	1	1	0.025	0.1	0.9	8	512	1500	10 Machine	15	15000	28120
18	18	1	1	0.025	0.01	0.7	4	512	1000	10 Machine	15	10000	37500
50	50	1	1	0.025	0.01	0.7	4	512	1000	10 Machine	15	10000	37500
82	82	1	1	0.025	0.01	0.7	4	512	1000	10 Machine	15	10000	37500
114	114	1	1	0.025	0.01	0.7	4	512	1000	10 Machine	15	10000	37500
146	146	1	1	0.025	0.01	0.7	4	512	1000	10 Machine	15	10000	37500
1	1	1	1	0.001	0.01	0.7	4	64	1500	10 Machine	15	15000	56248
33	33	1	1	0.001	0.01	0.7	4	64	1500	10 Machine	15	15000	56248
65	65	1	1	0.001	0.01	0.7	4	64	1500	10 Machine	15	15000	56248
97	97	1	1	0.001	0.01	0.7	4	64	1500	10 Machine	15	15000	56248
129	129	1	1	0.001	0.01	0.7	4	64	1500	10 Machine	15	15000	56248

# of iterations	actual iterations	repli cate	trian run	test run	date comp	total train speed	training speed (s/it)	evaluation ap for bbox	evaluatio n for segm ap
3125	3122	1	52a3b1a9	4c2d1724	15-Sep	0:26:57	0.3846	81.9513	81.2436
3125	3122	2	f2d70fb1	66d90104	14-Sep	0:26:56	0.3839	81.8064	81.6913
3125	3122	3	64ef3e73	7adad063	13-Sep	0:26:48	0.3836	81.3931	80.7063
3125	3122	4	30e88beb	da6c2c21	14-Sep	0:26:57	0.384	81.8745	81.3585
3125	3117	5	37fb9b56	09e51a80	16-Sep	0:26:50	0.3845	81.8491	81.3752
4688	4685	1	1e8afd01	aa04aa1c	13-Sep	0:50:49	0.4672	84.437	81.5097
4688	4677	2	c32e78d5	d9482427	18-Sep	0:50:54	0.4682	85.0879	81.9605
4688	4677	3	1253c747	c3e51cc2	19-Sep	0:50:44	0.4664	84.7027	81.9138
4688	4677	4	f5b79e37	1ea8a16d	19-Sep	0:51:07	0.4707	85.1264	82.4894
4688	4677	5	ff2da0d8	e6393759	16-Sep	0:50:56	0.4694	84.8682	82.1044
6250	6247	1	b1abd4b0	3e9aeffd	13-Sep	0:38:25	0.2448	85.3209	82.7922
6250	6247	2	b67294c8	5bf9b50c	14-Sep	0:38:32	0.2454	84.6419	81.8791
6250	6247	3	16bc418d	ba8a95d5	15-Sep	0:38:19	0.2436	85.4943	82.9517
6250	6245	4	57722199	bf4abff4	19-Sep	0:38:14	0.243	84.7056	82.1273
6250	6245	5	5939a0f7	f4f7d858	16-Sep	0:38:26	0.245	85.1677	82.414
9375	9372	1	f5c069b4	9e1aef64	13-Sep	1:19:13	0.3842	84.683	81.7841
9375	9372	2	7f41afa2	1d4901f7	15-Sep	1:19:17	0.3842	84.8272	81.7649
9375	9365	3	e7162139	0d70e4ed	18-Sep	1:19:18	0.3841	84.0652	81.5478
9375	9365	4	c2b84e7a	6ca4f8b7	17-Sep	1:19:23	0.3849	84.3279	81.4018
9375	9365	5	afaadaa5	b9626e0b	16-Sep	1:19:13	0.3838	84.9312	82.1564
9375	9372	1	ee58e192	1d5e49e	13-Sep	1:02:54	0.2012	73.964	77.6278
9375	9369	2	33893562		18-Sep	1:02:42	0.2015	75.441	79.1735
9375	9369	3	da6e176e	e4437cce	18-Sep	1:03:13	0.2012	73.9274	76.7805
9375	9369	4	2c7241306e	d9a1cdfa	19-Sep	1:02:39	0.2016	75.8571	79.1281
9375	9371	5	f995db35	fa3dcbcd	16-Sep	1:02:32	0.2017	74.2017	77.3075
14062	14060	1	4332d2d6	3ed12576	13-Sep	2:35:29	0.4862	88.079	85.1725
14063	14053	2	9c766728	448db038	18-Sep	2:35:53	0.4879	87.6619	84.8076
14063	14053	3	1efc5597	db79cd97	18-Sep	2:35:59	0.4884	88.1448	85.4513
14063	14053	4	9c766728	63892a5d	18-Sep	2:35:53	0.4874	87.6619	84.8076
14063	14053	5	996ca387	750df8e2	17-Sep	2:35:17	0.4864	87.7098	85.3336
18750	18747	1	21872f70	d25a98d6	13-Sep	1:56:14	0.2501	86.0584	83.4736
18750	18747	2	59691a61	bf20a93b	15-Sep	0:38:09	0.2537	87.2719	84.6016
18750	18745	3	86837212	1a51d25d	18-Sep	1:57:11	0.2526	86.3265	84.1992
18750	18745	4	6db3025e	d099a436	18-Sep	1:56:24	0.2507	85.3177	82.7213
18750	18745	5	ac4ece70	b9304f77	16-Sep	1:57:09	0.253	88.3981	85.9288
28125	28122	1	b2d6394f	f1b27e15	13-Sep	2:56:41	0.2011	86.9796	83.5209
28125	28124	2	f7fc18f4	f7090efa	16-Sep	2:56:40	0.2015	86.7553	83.5899
28125	28121	3	76f52b7c	7b68eb82	18-Sep	2:57:10	0.2016	86.9401	83.5932
28125	28121	4	37b56072	c3b980d8	18-Sep	2:57:09	0.2015	86.4579	83.5132
28125	28123	5	b04ccbe9	ee24c181	17-Sep	2:56:42	0.2016	86.9825	83.7064
6250	6245	1	bfacccec	e8f1dd3a	26-Sep	1:20:21	0.5178	83.2874	82.2261
6250	6245	2	bc961b2c	d1975fdc	26-Sep	1:19:54	0.5154	83.0655	82.1363
6250	6245	3	08ae9d56	229d135d	26-Sep	1:19:49	0.5141	82.3672	81.678
6250	6245	4	ef5978bc	77aaaaad	26-Sep	1:20:12	0.5172	82.7635	81.9974

6250	6245	5 bf67ad39	0b7ea74b	26-Sep	1:20:25	0.5173	82.9481	82.1856
9375	9365	1 c2c0d7cf	d02b4f15	21-Sep	2:01:18	0.3859	82.2247	80.6245
9375	9365	2 fbd79fb7	6a36fb03	21-Sep	2:01:17	0.3854	82.4096	80.7626
9375	9365	3 a5f7848a	36725ba7	22-Sep	2:01:23	0.386	82.8284	81.1398
9375	9365	4 42a3a6d5	fca4a7d7	22-Sep	2:01:17	0.3862	82.4843	80.777
9375	9365	5 b74a63f5	ab48cb8a	22-Sep	2:01:31	0.3867	82.1454	80.4027
12500	12497	1 3b24f282	7ba0dc18	24-Sep	1:35:57	0.2002	82.6974	80.849
12500	12497	2 847427e9	44679a38	24-Sep	1:35:52	0.1999	82.3365	80.5398
12500	12497	3 2386f816	df5723fb	25-Sep	1:35:35	0.2002	82.7873	80.962
12500	12497	4 84acc825	438cbb3f	25-Sep	1:35:58	0.2002	82.6038	80.7523
12500	12497	5 b27b3515	f9ea007e	25-Sep	1:35:27	0.1999	82.8105	80.9619
18750	18741	1 13be02ea	285289ff	25-Sep	3:59:27	0.5149	84.2499	82.3622
18750	18741	2 900e80ac	2b8533eb	25-Sep	3:59:06	0.5139	84.01	81.9949
18750	18741	3 69d538cf	528f015c	25-Sep	3:57:38	0.5101	83.8175	81.8878,8
18750	18741	4 cf15bcf0	ca158a23	25-Sep	3:58:07	0.5119	83.8561	81.8879
18750	18741	5 e70c0676	7c409f89	25-Sep	3:59:00	0.5139	84.3429	82.4747
18750	18745	1 74976d5a	be9f7b14	28-Sep	3:21:58	0.2677	83.8357	82.8656
18750	18745	2 edbb94c7	50a18333	28-Sep	3:23:34	0.2704	84.2131	82.9433
18750	18745	3 f192766e	008bb19a	28-Sep	3:22:10	0.2669	84.0109	82.8635
18750	18745	4 d5e2ced6	00460b03	29-Sep	3:22:13	0.2683	84.8666	83.5316
18750	18745	5 0069c6fe	3c690cfc	29-Sep	3:22:01	0.2671	84.5283	83.1975
28125	28117	1 0b5aa987	23ea5805	29-Sep	5:59:46	0.3829	84.6649	83.5274
28125	28117	2 96a065c6	443342fe	29-Sep	5:59:54	0.3836	84.687	83.4155
28125	28117	3 e890e8e6	34e97e40	30-Sep	5:59:43	0.3836	84.5221	83.3966
28125	28117	4 5c9dc486	39acaf92	30-Sep	6:00:14	0.3836	85.0852	83.7672
28125	28117	5 aaa9d92e	a6dc3252	30-Sep	6:00:07	0.3835	84.8284	83.5729
37500	37497	1 27d46963	5f614f74	28-Sep	5:02:40	0.2002	76.0954	79.1156
37500	37497	2 b88f35e5	1b85182a	28-Sep	5:02:37	0.1999	75.8143	78.7169
37500	37497	3 8bd9379d	21f8fc97	26-Sep	5:03:25	0.1998	75.6309	78.2559
37500	37497	4 ab90aab3	8655d346	27-Sep	5:04:11	0.2002	76.5386	79.3185
37500	37497	5 6c7547d7	2168d304	27-Sep	5:04:43	0.2	75.7272	78.4444
56250	56245	1 4169834b	d5e94f54	23-Sep	9:55:08	0.27	85.9365	83.721
56250	56245	2 7d4517b5	35b0b33f	23-Sep	9:54:46	0.2702	86.1357	83.9059
56250	56245	3 714691ef	320ac347	23-Sep	9:53:17	0.2695	85.8059	83.5412
56250	56245	4 47430eb9	eeb353a3	24-Sep	9:51:55	0.2683	85.6299	83.4052
56250	56245	5 30f5eef0	7d88fab5	24-Sep	9:51:29	0.2675	85.8069	83.554
3125	3117	1 c1df1806	8f79839c	7-Oct	0:33:32	0.5155	90.3983	90.472
3125	3117	2 206f5b8f	084ef12c	7-Oct	0:33:33	0.5149	91.0376	91.0591
3125	3117	3 ddf664e1	65f312a9	7-Oct	0:33:27	0.5132	90.9269	90.7106
3125	3117	4 a8e0372c	7d295935	7-Oct	0:33:35	0.5155	91.0902	91.1573
3125	3117	5 706d9306	c29c36d2	7-Oct	0:33:46	0.5189	91.7119	91.5465
4688	4677	1 3c486814	e08ebd54	8-Oct	0:44:22	0.382	89.727	89.2048
4688	4677	2 efe300d3	2d74de8d	8-Oct	0:44:20	0.3819	89.9235	89.585
4688	4677	3 15a11657	f4b13186	8-Oct	0:44:25	0.3821	90.0382	89.5151
4688	4677	4 85b0cf70	d6ec1614	8-Oct	0:44:20	0.3816	89.9482	89.4271
4688	4677	5 d334826d	cf853e0a	8-Oct	0:44:18	0.3819	89.9213	89.6154
6250	6245	1 9fdcd0fc	d6be73a2	7-Oct	0:34:02	0.2001	89.6657	89.7584

6250	6245	2	526e104f	7a750aee	7-Oct	0:34:03	0.2001	88.7795	88.8986
6250	6245	3	859fbf60	3dfd678a	7-Oct	0:34:00	0.1997	89.5358	89.8034
6250	6245	4	895e07c2	8a7973be	8-Oct	0:33:57	0.1995	88.9715	88.684
6250	6245	5	e42229d9	da6dfdada	8-Oct	0:34:02	0.1996	89.5387	89.3427
9375	9365	1	1e27cb3d	79f562a2	8-Oct	1:37:22	0.4965	90.9969	90.3541
9375	9365	2	0298544c	abc949b2	8-Oct	1:38:32	0.5039	90.7709	90.2602
9375	9365	3	c62384f7	a2a798a8	8-Oct	1:37:47	0.4992	90.7097	90.5499
9375	9365	4	b2cc41a2	b8f02821	8-Oct	1:37:26	0.4972	90.9454	90.6846
9375	9365	5	ae52236d	e68f1730	8-Oct	1:37:14	0.4952	91.061	90.4546
9375	9369	1	76073ab7	8805eace	7-Oct	1:11:17	0.2682	90.0677	90.9809
9375	9369	2	b604a06a	4bca69a5	6-Oct	1:10:51	0.2673	89.5545	90.2195
9375	9369	3	df7e3f48	af1e369c	6-Oct	1:11:18	0.2691	90.0656	90.7292
9375	9369	4	1972e545	ad256d3f	7-Oct	1:10:43	0.2658	89.6138	90.3066
9375	9369	5	afdf5b40	90cd8fbf	7-Oct	1:10:31	0.2652	89.6736	90.5919
14063	14053	1	8011b77e	4b60cfe4	7-Oct	2:13:27	0.3828	89.937	90.9437
14063	14053	2	b37dd4d0	b3f0562d	7-Oct	2:13:55	0.3839	88.8463	90.0347
14063	14053	3	9c44ad1e	e5282945	7-Oct	2:13:39	0.3834	89.1435	90.4133
14063	14053	4	f6e3e39b	0485fdda	7-Oct	2:13:44	0.3836	89.3069	90.4673
14063	14053	5	da10de1f	0be4cbf4	7-Oct	2:13:47	0.3834	89.2628	90.4223
18750	18745	1	a525b6f7	9b5790a0	6-Oct	1:44:11	0.2002	87.9288	89.3824
18750	18745	2	87642d2d	7557f1c8	6-Oct	1:44:11	0.2002	87.9288	89.3824
18750	18745	3	514e15ad	8c0fce5f	6-Oct	1:43:39	0.1999	88.9035	90.2857
18750	18745	4	87642d2d	7557f1c8	6-Oct	1:43:53	0.2	88.3448	89.9362
18750	18745	5	60897385	4122529e	7-Oct	1:44:05	0.1999	89.9625	88.4161
28125	28121	1	f83a8286	09b97383	8-Oct	3:27:00	0.2615	91.9107	91.3235
28125	28121	2	8982fd31	f08d194b	8-Oct	3:26:20	0.2601	91.9141	91.38
28125	28121	3	438607cd	a399e145	8-Oct	3:29:10	0.2651	91.666	91.1129
28125	28121	4	aff3fea3	7e8e82e7	9-Oct	3:27:41	0.2629	92.116	91.5821
28125	28121	5	7d12308d	295a60f8	9-Oct	3:29:12	0.2661	91.821	91.0602
6250	6245	1	f8c2416c	82bb377d	13-Oct	1:10:56	0.3853	67.3888	70.4276
6250	6245	2	e1a8250b	efdb0971	13-Oct	1:10:42	0.385	67.743	70.8915
6250	6245	3	49069e8d	f6132fa7	13-Oct	1:10:51	0.3847	67.6358	70.6867
6250	6245	4	747636ab	817724a2	13-Oct	1:10:50	0.385	67.0805	70.0758
6250	6245	5	bec7154	02c52089	13-Oct	1:11:01	0.3841	66.821	69.9444
9375	9365	1	138579e6	0e1f1ff3	16-Oct	2:24:13	0.5048	73.1892	72.7913
9375	9365	2	276abead	192e5d74	16-Oct	2:24:41	0.5068	72.9644	72.6251
9375	9365	3	44467f2a	526c5372	16-Oct	2:24:36	0.5065	73.5811	73.2339
9375	9365	4	823a9sd	23432dw	16-Oct	2:24:34	0.50603	73.2449	72.8834
9375	9365	5	82cc20d7	b09d0543	16-Oct	2:24:46	0.5073	73.1633	72.7567
12500	12497	1	b26d7e39	790c72c1	13-Oct	1:53:21	0.2664	74.6744	74.0732
12500	12497	2	9d4f52a3	3aa2cd92	13-Oct	1:52:52	0.2652	74.9057	74.4715
12500	12497	3	b108cf3a	af47f1b1	14-Oct	1:53:18	0.2661	74.5678	74.0653
12500	12497	4	ef0f9bb8	2c331926	14-Oct	1:52:52	0.2655	75.1681	74.5686
12500	12497	5	766fec82	aae99014	14-Oct	1:52:56	0.2658	74.866	74.3165
18750	18741	1	140dddef	dd38bd92	14-Oct	3:24:31	0.3837	75.7553	75.111
18750	18741	2	f176423b	ba03f3b3	14-Oct	3:24:39	0.3839	75.7897	75.1187
18750	18741	3	0520d807	897d4f92	14-Oct	3:25:11	0.3845	75.5622	74.8959

18750	18741	4	e41b7318	7f7e1ce1	14-Oct	3:24:46	0.3843	75.7273	75.0437
18750	18741	5	f56ed5be	7c948e32	14-Oct	3:24:50	0.3843	75.5854	75.0307
18750	18745	1	0cd382bb	7b2f8053	16-Oct	3:31:43	0.2008	67.4559	70.2364
18750	18745	2	84bd6717	49c19635	17-Oct	3:30:39	0.2007	68.3136	70.9479
18750	18745	3	9189aebb	8d2395a5	17-Oct	3:30:43	0.2004	68.5824	71.2159
18750	18745	4	b93f5f22	b91aafb5	17-Oct	3:30:37	0.2002	67.578	70.3115
18750	18745	5	906cf8ce	d97736d8	17-Oct	3:31:15	0.2	68.2449	70.8295
28125	28117	1	61539401	324ef599	17-Oct	7:12:40	0.5303	79.4272	78.7739
28125	28117	2	3e7b365b	5e3b967d	17-Oct	7:13:15	0.531	79.1618	78.5659
28125	28117	3	9ecb4fb2	76b6e9c1	18-Oct	7:12:52	0.5305	79.5093	78.8413
28125	28117	4	89bd109c	41be7541	18-Oct	7:13:36	0.532	79.5489	78.8031
28125	28117	5	fff52d9f	3f47a45c	18-Oct	7:12:14	0.5295	79.2706	78.6014
37500	37497	1	29d4b3c7	9b38ec82	15-Oct	5:43:45	0.2747	75.7758	76.1428
37500	37497	2	25321089	4ca19a3b	15-Oct	5:43:42	0.2741	75.8302	76.416
37500	37497	3	5fc58009	8wer82	15-Oct	5:43:32	0.2745	75.25	75.7213
37500	37497	4	5a793071	bc312d6f	15-Oct	5:44:52	0.2753	75.0709	75.7429
37500	37497	5	6d7360e1	292f35bb	15-Oct	5:43:06	0.2737	75.2531	75.6996
56250	56245	1	62e18b45	ace3df26	19-Oct	9:16:47	0.2003	77.4471	76.8365
56250	56245	2	0f596edd	5ec7885b	19-Oct	9:15:45	0.2003	77.4541	76.8095
56250	56245	3	4630f8c6	64cf5016	20-Oct	9:16:38	0.2003	77.343	76.6999
56250	56245	4	24daa859	00f39fdd	20-Oct	9:15:23	0.2003	77.3784	76.7782
56250	56245	5	c7e2eb68	fba49c4b	20-Oct	9:16:46	0.2004	77.4514	76.8338

test -evalua	test -evalua	generate	old bbox	old seg	Mean bbox AP	Mean seg AP
80.064142	79.618069	517c2d08b4				
80.728217	79.889659	517c2d08b4				
80.282345	79.559707	517c2d08b4				
80.692228	79.938413	517c2d08b4				
80.542776	79.809744	517c2d08b4			80.5058003	79.76929
84.788839	81.639965	517c2d08b4				
84.755585	81.878786	517c2d08b4				
84.692047	81.642618	517c2d08b4				
84.601034	81.679538	517c2d08b4				
84.697819	81.64358	517c2d08b4			84.7070648	81.6969
84.067353	81.224967	517c2d08b4				
83.560661	80.841702	517c2d08b4				
84.460426	81.738909	517c2d08b4				
84.003976	81.631902	517c2d08b4				
84.001062	81.133002	517c2d08b4			84.0186956	81.3141
83.387035	80.574358	517c2d08b4				
84.210362	81.40972	517c2d08b4				
83.205387	80.201006	517c2d08b4				
83.882573	80.910355	517c2d08b4				
83.504019	80.522007	517c2d08b4			83.6378752	80.72349
73.952718	77.452223	517c2d08b4				
73.877147	77.292761	517c2d08b4				
73.45902	76.719968	517c2d08b4				
74.53778	77.639089	517c2d08b4				
73.776163	76.901354	517c2d08b4			73.9205656	77.20108
87.366373	84.454218	517c2d08b4				
87.550956	84.568648	517c2d08b4				
87.72386	84.86915	517c2d08b4				
87.550956	84.568648	517c2d08b4				
87.416445	84.414103	517c2d08b4			87.521718	84.57495
85.841767	83.23155	517c2d08b4				
86.570535	83.587306	517c2d08b4				
86.666298	83.948468	517c2d08b4				
86.12996	83.489727	517c2d08b4				
86.431206	83.89342	517c2d08b4			86.3279532	83.63009
85.902926	82.342668	517c2d08b4				
86.092127	82.71589	517c2d08b4				
85.419589	82.060913	517c2d08b4				
86.097326	82.726458	517c2d08b4				
85.668385	82.295418	517c2d08b4			85.8360706	82.42827
84.031606	83.176972	384376e1ea				
83.703969	83.041443	384376e1ea				
83.384947	82.51092	384376e1ea				
83.892438	83.154582	384376e1ea				

83.901318	82.918802	384376e1ea	83.7828556	82.96054
82.499886	81.031871	384376e1ea		
82.407989	80.84613	384376e1ea		
82.605513	80.799358	384376e1ea		
82.156397	80.520067	384376e1ea		
81.928708	80.326137	384376e1ea	82.3196986	80.70471
83.350396	81.695595	384376e1ea		
83.535931	81.857687	384376e1ea		
83.547193	81.848815	384376e1ea		
83.26856	81.526061	384376e1ea		
83.734789	81.919738	384376e1ea	83.4873738	81.76958
84.443655	82.518603	384376e1ea		
84.46505	82.300166	384376e1ea		
84.170536	82.166655	384376e1ea		
84.540108	82.489505	384376e1ea		
84.484098	82.381168	384376e1ea	84.4206894	82.37122
85.005182	83.993649	384376e1ea		
85.278851	83.904995	384376e1ea		
84.693168	83.670061	384376e1ea		
85.531647	84.253621	384376e1ea		
85.100204	83.988508	384376e1ea	85.1218104	83.96217
85.241657	84.005554	384376e1ea		
86.04327	84.604095	384376e1ea		
85.332822	84.091432	384376e1ea		
85.678437	84.316801	384376e1ea		
85.47221	84.070177	384376e1ea	85.5536792	84.21761
76.494298	79.324535	384376e1ea		
76.364659	79.548473	384376e1ea		
76.690152	79.58851	384376e1ea		
77.281206	79.718461	384376e1ea		
76.977334	79.554706	384376e1ea	76.7615298	79.54694
85.52405	83.498419	384376e1ea		
85.23501	83.274805	384376e1ea		
85.572327	83.439531	384376e1ea		
85.294712	83.416366	384376e1ea		
85.590772	83.395202	384376e1ea	85.4433741	83.40486
90.888836	90.857617	d166eefb		
90.715981	90.801554	d166eefb		
91.041358	91.233178	d166eefb		
90.728606	90.99272	d166eefb		
91.097292	91.116958	d166eefb	90.912949	91.05484
89.902591	89.449338	d166eefb		
89.776665	89.635904	d166eefb		
89.893385	89.839228	d166eefb		
89.756244	89.396566	d166eefb		
89.800006	89.496003	d166eefb	89.8257782	89.56341
88.770723	88.893127	d166eefb		

89.358026	89.681952	d166eefb		
89.208994	89.610425	d166eefb		
89.312151	89.1328	d166eefb		
89.055389	88.809404	d166eefb	89.1410566	89.22554
90.827408	90.655609	d166eefb		
90.648975	90.349647	d166eefb		
90.645094	90.331276	d166eefb		
90.512054	90.266117	d166eefb		
90.908906	90.786166	d166eefb	90.6787573	90.4333
89.478721	90.518777	d166eefb		
90.122465	91.065323	d166eefb		
89.975671	90.688985	d166eefb		
89.685866	90.659942	d166eefb		
89.78532	90.743517	d166eefb	89.8096086	90.73531
89.546228	90.502653	d166eefb		
88.992355	90.31314	d166eefb		
88.728443	90.1718	d166eefb		
89.585245	90.758648	d166eefb		
88.625125	89.812576	d166eefb	89.0954792	90.31176
87.805415	89.441666	d166eefb		
87.799488	89.231289	d166eefb		
88.248662	89.550354	d166eefb		
87.799488	89.231289	d166eefb		
87.86307	89.481107	d166eefb	87.9032246	89.38714
91.556585	91.267007	d166eefb		
91.699854	91.295024	d166eefb		
91.503173	91.121793	d166eefb		
91.695148	91.211254	d166eefb		
91.814343	91.430164	d166eefb	91.6781295	91.26456
71.588193	75.322944	f50fb95d	67.536	70.3953
71.951982	75.588467		67.879	70.6434
71.719257	75.376049		67.66	70.4449
71.245928	75.000149		67.213	70.0936
70.561844	74.269285		66.568	69.4105 71.4134406 75.11138
78.535797	77.818243		73.398	72.7273
78.396941	77.647634		73.268	72.5679
78.633685	77.987041		73.489	72.8851
78.522141	77.817639		73.385	72.7268
78.496705	77.816171		73.361	72.7254 78.5170538 77.81735
80.783205	79.916028		74.799	73.9963
80.813163	79.933711		74.827	74.0127
80.798427	80.04833		74.813	74.1188
80.82775	79.841143		74.841	73.927
80.766897	79.842294		74.784	73.9281 80.7978884 79.9163
81.779012	81.393671		75.721	74.6731
82.057009	81.749115		75.979	74.9992
82.596142	82.237822		76.478	75.4475

82.148404	81.847313	76.063	75.0893		
81.970449	81.608614	75.899	74.8703	82.1930009	81.86072
71.564581	75.452482	67.514	69.8634		
72.727447	76.724151	68.611	71.0409		
73.123675	76.905593	68.985	71.2089		
71.51923	75.490006	67.471	69.8982		
72.905486	76.681349	68.779	71.0012	72.3680838	76.25072
86.287089	85.491035	79.895	79.1584		
86.166971	85.4378	79.784	79.1091		
86.684048	85.731234	80.263	79.3808		
86.286935	85.310389	79.895	78.9911		
86.091582	85.292304	79.714	78.9744	86.3033249	85.45255
81.783565	81.756626	75.726	75.7006		
81.325714	81.510179	75.302	75.4724		
81.970547	82.200645	75.899	76.1117		
81.228042	81.627268	75.211	75.5808		
82.713052	82.774022	76.586	76.6426	81.8041839	81.97375
84.123155	83.169134	77.892	77.0085		
83.962887	83.050647	77.743	76.8987		
84.013593	83.082121	77.79	76.9279		
83.987379	82.833295	77.766	76.6975		
84.533262	83.600098	78.272	77.4075	84.1242804	83.14154

APPENDIX F

2^k Factorial Design Minitab Results

Bounding Box Minitab Regression Results: Bounding Box AP vs. 9 Factors and Interactions

■ CORRECT LABEL FULL DESING

Factorial Regression: test -evaluation ap for bbox versus LR, LRD, LR Decline, Batch, ROi, NumImagesperClass, NumClasses, Dat

Coded Coefficients

Term	Effect	Coef	SE Coef	T-Value	P-Value	VIF
Constant		84.0560	0.0248	3383.12	0.000	
LR	-1.9816	-0.9908	0.0248	-39.88	0.000	1.00
LRD	1.6096	0.8048	0.0248	32.39	0.000	1.00
LR Decline	0.1348	0.0674	0.0248	2.71	0.008	1.00
Batch	0.7972	0.3986	0.0248	16.04	0.000	1.00
ROi	3.3671	1.6835	0.0248	67.76	0.000	1.00
NumImagesperClass	0.9031	0.4515	0.0248	18.17	0.000	1.00
NumClasses	-5.0709	-2.5355	0.0248	-102.05	0.000	1.00
Dataset	1.4467	0.7233	0.0248	29.11	0.000	1.00
epoch	3.0386	1.5193	0.0248	61.15	0.000	1.00
LR*LRD	0.9935	0.4968	0.0248	19.99	0.000	1.00
LR*LR Decline	-0.0375	-0.0188	0.0248	-0.76	0.451	1.00
LR*Batch	1.8291	0.9145	0.0248	36.81	0.000	1.00
LR*ROi	3.3939	1.6970	0.0248	68.30	0.000	1.00
LR*NumImagesperClass	0.3900	0.1950	0.0248	7.85	0.000	1.00
LR*NumClasses	-0.2823	-0.1412	0.0248	-5.68	0.000	1.00
LR*Dataset	-0.1792	-0.0896	0.0248	-3.61	0.000	1.00
LR*epoch	1.1486	0.5743	0.0248	23.12	0.000	1.00
LRD*LR Decline	-5.1283	-2.5641	0.0248	-103.20	0.000	1.00
LRD*Batch	0.5505	0.2752	0.0248	11.08	0.000	1.00
LRD*ROi	-0.5320	-0.2660	0.0248	-10.71	0.000	1.00
LRD*Dataset	-0.7042	-0.3521	0.0248	-14.17	0.000	1.00
LRD*epoch	-0.2331	-0.1165	0.0248	-4.69	0.000	1.00
LR Decline*Batch	0.8283	0.4141	0.0248	16.67	0.000	1.00
LR Decline*ROi	0.3563	0.1781	0.0248	7.17	0.000	1.00
LR Decline*epoch	-0.6389	-0.3194	0.0248	-12.86	0.000	1.00
Batch*ROi	-0.5624	-0.2812	0.0248	-11.32	0.000	1.00
ROi*NumImagesperClass	-0.1128	-0.0564	0.0248	-2.27	0.025	1.00
ROi*NumClasses	0.1396	0.0698	0.0248	2.81	0.006	1.00
ROi*Dataset	-0.3036	-0.1518	0.0248	-6.11	0.000	1.00
ROi*epoch	-0.9719	-0.4859	0.0248	-19.56	0.000	1.00
LR*ROi*NumImagesperClass	0.3064	0.1532	0.0248	6.17	0.000	1.00

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.314276	99.70%	99.62%	99.53%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	31	4157.45	134.11	1357.82	0.000
Linear	9	2254.57	250.51	2536.28	0.000
LR	1	157.07	157.07	1590.24	0.000
LRD	1	103.63	103.63	1049.34	0.000
LR Decline	1	0.73	0.73	7.36	0.008
Batch	1	25.42	25.42	257.35	0.000
ROi	1	453.48	453.48	4591.31	0.000
NumImagesperClass	1	32.62	32.62	330.29	0.000
NumClasses	1	1028.56	1028.56	10413.76	0.000
Dataset	1	83.72	83.72	847.60	0.000
epoch	1	369.33	369.33	3739.35	0.000
2-Way Interactions	21	1899.13	90.43	915.61	0.000
LR*LRD	1	39.48	39.48	399.76	0.000
LR*LR Decline	1	0.06	0.06	0.57	0.451
LR*Batch	1	133.82	133.82	1354.89	0.000
LR*ROi	1	460.75	460.75	4664.91	0.000
LR*NumImagesperClass	1	6.08	6.08	61.60	0.000
LR*NumClasses	1	3.19	3.19	32.28	0.000
LR*Dataset	1	1.29	1.29	13.01	0.000
LR*epoch	1	52.78	52.78	534.33	0.000
LRD*LR Decline	1	1051.98	1051.98	10650.81	0.000
LRD*Batch	1	12.12	12.12	122.71	0.000
LRD*ROi	1	11.32	11.32	114.61	0.000
LRD*Dataset	1	19.83	19.83	200.82	0.000
LRD*epoch	1	2.17	2.17	22.00	0.000
LR Decline*Batch	1	27.44	27.44	277.84	0.000
LR Decline*ROi	1	5.08	5.08	51.40	0.000
LR Decline*epoch	1	16.33	16.33	165.29	0.000
Batch*ROi	1	12.65	12.65	128.08	0.000
ROi*NumImagesperClass	1	0.51	0.51	5.15	0.025
ROi*NumClasses	1	0.78	0.78	7.89	0.006
ROi*Dataset	1	3.69	3.69	37.33	0.000
ROi*epoch	1	37.78	37.78	382.52	0.000
3-Way Interactions	1	3.76	3.76	38.03	0.000
LR*ROi*NumImagesperClass	1	3.76	3.76	38.03	0.000
Error	128	12.64	0.10		

Total 159 4170.09

Regression Equation in Uncoded Units

test -evaluation ap for bbox = 87.743 - 603.2 LR + 0.36 LRD + 2.389 LR Decline - 0.2835 Batch
+ 0.012893 ROI + 0.002105 NumImagesperClass
- 0.9889 NumClasses + 1.4460 Dataset + 0.3329 epoch
+ 919.9 LR*LRD - 1.56 LR*LR Decline + 38.11 LR*Batch
+ 0.3463 LR*ROI - 0.0007 LR*NumImagesperClass
- 4.706 LR*NumClasses - 7.47 LR*Dataset + 9.572 LR*epoch
- 56.981 LRD*LR Decline + 3.058 LRD*Batch - 0.02639 LRD*ROI
- 7.824 LRD*Dataset - 0.518 LRD*epoch
+ 0.2071 LR Decline*Batch + 0.000795 LR Decline*ROI
- 0.06389 LR Decline*epoch - 0.000628 Batch*ROI
- 0.000004 ROI*NumImagesperClass + 0.000125 ROI*NumClasses
- 0.000678 ROI*Dataset - 0.000434 ROI*epoch
+ 0.000228 LR*ROI*NumImagesperClass

Fits and Diagnostics for Unusual Observations

Obs	test		Fit	Resid	Std Resid	
	-evaluation	ap for bbox				
8	71.565	72.368	-0.804	-2.86	R	
41	84.210	83.638	0.572	2.04	R	
72	73.124	72.368	0.756	2.69	R	
98	74.538	73.921	0.617	2.20	R	
104	71.519	72.368	-0.849	-3.02	R	
114	81.228	81.804	-0.576	-2.05	R	
138	70.562	71.413	-0.852	-3.03	R	
146	82.713	81.804	0.909	3.23	R	

R Large residual

Alias Structure (up to order 3)

Factor	Name
A	LR
B	LRD
C	LR Decline
D	Batch
E	ROI
F	NumImagesperClass
G	NumClasses
H	Dataset
J	epoch
Aliases	
I	
A + BFG + CFH + DFJ	
B + AFG + CGH + DGJ	
C + AFH + BGH + DHJ	
D + AFJ + BGJ + CHJ	
E	
F + ABG + ACH + ADJ	
G + ABF + BCH + BDJ	
H + ACF + BCG + CDJ	
J + ADF + BDG + CDH	
AB + FG + CEJ + DEH	
AC + FH + BEJ + DEG	
AD + FJ + BEH + CEG	
AE + BCJ + BDH + CDG + GHJ	
AF + BG + CH + DJ	
AG + BF + CDE + EHJ	
AH + CF + BDE + EGJ	
AJ + DF + BCE + EGH	
BC + GH + AEJ + DEF	
BD + GJ + AEH + CEF	
BE + ACJ + ADH + CDF + FHJ	
BH + CG + ADE + EFJ	
BJ + DG + ACE + EFH	
CD + HJ + AEG + BEF	
CE + ABJ + ADG + BDF + FGJ	
CJ + DH + ABE + EFG	
DE + ABH + ACG + BCF + FGH	
EF + BCD + BHJ + CGJ + DGH	
EG + ACD + AHJ + CFJ + DFH	
EH + ABD + AGJ + BFJ + DFG	
EJ + ABC + AGH + BFH + CFG	
AEF + BEG + CEH + DEJ	

Factorial Plots for test -evaluation ap for bbox

Segmentation Minitab Regression Results: Segmentation AP vs. 9 Factors and Interactions

CORRECT LABEL FULL DESING

Factorial Regression: test -evaluation for segm ap versus LR, LRD, LR Decline, Batch, ROi, NumImagesperClass, NumClasses, Dataset, epoch

Coded Coefficients

Term	Effect	Coef	SE Coef	T-Value	P-Value	VIF
Constant		83.5523	0.0235	3559.09	0.000	
LR	-0.0946	-0.0473	0.0235	-2.01	0.046	1.00
LRD	0.9507	0.4754	0.0235	20.25	0.000	1.00
LR Decline	0.1796	0.0898	0.0235	3.82	0.000	1.00
Batch	0.2098	0.1049	0.0235	4.47	0.000	1.00
ROi	1.9646	0.9823	0.0235	41.84	0.000	1.00
NumImagesperClass	0.7372	0.3686	0.0235	15.70	0.000	1.00
NumClasses	-4.5578	-2.2789	0.0235	-97.07	0.000	1.00
Dataset	3.3208	1.6604	0.0235	70.73	0.000	1.00
epoch	2.2305	1.1152	0.0235	47.51	0.000	1.00
LR*LRD	0.4553	0.2277	0.0235	9.70	0.000	1.00
LR*LR Decline	-0.0267	-0.0134	0.0235	-0.57	0.570	1.00
LR*Batch	1.1284	0.5642	0.0235	24.03	0.000	1.00
LR*ROi	2.0979	1.0489	0.0235	44.68	0.000	1.00
LR*NumImagesperClass	0.4294	0.2147	0.0235	9.15	0.000	1.00
LR*NumClasses	-0.0833	-0.0416	0.0235	-1.77	0.079	1.00
LR*Dataset	-0.2750	-0.1375	0.0235	-5.86	0.000	1.00
LR*epoch	0.5333	0.2667	0.0235	11.36	0.000	1.00
LRD*LR Decline	-5.5085	-2.7542	0.0235	-117.32	0.000	1.00
LRD*Batch	0.6931	0.3466	0.0235	14.76	0.000	1.00
LRD*ROi	-0.1962	-0.0981	0.0235	-4.18	0.000	1.00
LRD*Dataset	-0.2253	-0.1127	0.0235	-4.80	0.000	1.00
LRD*epoch	-0.1559	-0.0779	0.0235	-3.32	0.001	1.00
LR Decline*Batch	0.7898	0.3949	0.0235	16.82	0.000	1.00
LR Decline*ROi	0.3199	0.1599	0.0235	6.81	0.000	1.00
LR Decline*epoch	-0.2596	-0.1298	0.0235	-5.53	0.000	1.00
Batch*ROi	-0.1910	-0.0955	0.0235	-4.07	0.000	1.00
ROi*NumImagesperClass	-0.0791	-0.0395	0.0235	-1.68	0.095	1.00
ROi*NumClasses	-0.0467	-0.0234	0.0235	-0.99	0.322	1.00
ROi*Dataset	-0.2304	-0.1152	0.0235	-4.91	0.000	1.00
ROi*epoch	-0.5121	-0.2560	0.0235	-10.91	0.000	1.00
LR*ROi*NumImagesperClass	0.2025	0.1013	0.0235	4.31	0.000	1.00

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.296947	99.65%	99.57%	99.46%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	31	3229.36	104.17	1181.40	0.000
Linear	9	1686.73	187.41	2125.43	0.000
LR	1	0.36	0.36	4.06	0.046
LRD	1	36.15	36.15	410.01	0.000
LR Decline	1	1.29	1.29	14.63	0.000
Batch	1	1.76	1.76	19.96	0.000
ROi	1	154.39	154.39	1750.92	0.000
NumImagesperClass	1	21.74	21.74	246.53	0.000
NumClasses	1	830.93	830.93	9423.39	0.000
Dataset	1	441.11	441.11	5002.57	0.000
epoch	1	199.00	199.00	2256.77	0.000
2-Way Interactions	21	1540.98	73.38	832.19	0.000
LR*LRD	1	8.29	8.29	94.04	0.000
LR*LR Decline	1	0.03	0.03	0.32	0.570
LR*Batch	1	50.93	50.93	577.56	0.000
LR*ROi	1	176.04	176.04	1996.45	0.000
LR*NumImagesperClass	1	7.38	7.38	83.64	0.000
LR*NumClasses	1	0.28	0.28	3.14	0.079
LR*Dataset	1	3.03	3.03	34.31	0.000
LR*epoch	1	11.38	11.38	129.02	0.000
LRD*LR Decline	1	1213.73	1213.73	13764.64	0.000
LRD*Batch	1	19.22	19.22	217.93	0.000
LRD*ROi	1	1.54	1.54	17.46	0.000
LRD*Dataset	1	2.03	2.03	23.03	0.000
LRD*epoch	1	0.97	0.97	11.02	0.001
LR Decline*Batch	1	24.95	24.95	282.96	0.000
LR Decline*ROi	1	4.09	4.09	46.41	0.000
LR Decline*epoch	1	2.70	2.70	30.58	0.000
Batch*ROi	1	1.46	1.46	16.55	0.000
ROi*NumImagesperClass	1	0.25	0.25	2.84	0.095
ROi*NumClasses	1	0.09	0.09	0.99	0.322
ROi*Dataset	1	2.12	2.12	24.07	0.000
ROi*epoch	1	10.49	10.49	118.94	0.000
3-Way Interactions	1	1.64	1.64	18.61	0.000
LR*ROi*NumImagesperClass	1	1.64	1.64	18.61	0.000
Error	128	11.29	0.09		

Total 159 3240.64

Regression Equation in Uncoded Units

test -evaluation for segm ap = 88.259 - 349.8 LR - 11.75 LRD + 2.340 LR Decline
- 0.4035 Batch + 0.007057 ROI + 0.001312 NumImagesperClass
- 0.8815 NumClasses + 2.0952 Dataset + 0.2502 epoch
+ 421.6 LR*LRD - 1.11 LR*LR Decline + 23.507 LR*Batch
+ 0.2019 LR*ROI + 0.0082 LR*NumImagesperClass
- 1.388 LR*NumClasses - 11.46 LR*Dataset + 4.444 LR*epoch
- 61.205 LRD*LR Decline + 3.851 LRD*Batch - 0.00973 LRD*ROI
- 2.504 LRD*Dataset - 0.346 LRD*epoch
+ 0.1974 LR Decline*Batch + 0.000714 LR Decline*ROI
- 0.02596 LR Decline*epoch - 0.000213 Batch*ROI
- 0.000003 ROI*NumImagesperClass - 0.000042 ROI*NumClasses
- 0.000514 ROI*Dataset - 0.000229 ROI*epoch
+ 0.000151 LR*ROI*NumImagesperClass

Fits and Diagnostics for Unusual Observations

Obs	test		Fit	Resid	Std Resid	
	-evaluation	for segm ap				
8	75.452	76.251	-0.798	-3.01	R	
41	81.410	80.723	0.686	2.58	R	
72	76.906	76.251	0.655	2.47	R	
104	75.490	76.251	-0.761	-2.86	R	
138	74.269	75.111	-0.842	-3.17	R	
146	82.774	81.974	0.800	3.01	R	

R Large residual

Alias Structure (up to order 3)

Factor	Name
A	LR
B	LRD
C	LR Decline
D	Batch
E	ROI
F	NumImagesperClass
G	NumClasses
H	Dataset
J	epoch
Aliases	
I	
A + BFG + CFH + DFJ	
B + AFG + CGH + DGJ	
C + AFH + BGH + DHJ	
D + AFJ + BGJ + CHJ	
E	
F + ABG + ACH + ADJ	
G + ABF + BCH + BDJ	
H + ACF + BCG + CDJ	
J + ADF + BDG + CDH	
AB + FG + CEJ + DEH	
AC + FH + BEJ + DEG	
AD + FJ + BEH + CEG	
AE + BCJ + BDH + CDG + GHJ	
AF + BG + CH + DJ	
AG + BF + CDE + EHJ	
AH + CF + BDE + EGJ	
AJ + DF + BCE + EGH	
BC + GH + AEJ + DEF	
BD + GJ + AEH + CEF	
BE + ACJ + ADH + CDF + FHJ	
BH + CG + ADE + EFJ	
BJ + DG + ACE + EFH	
CD + HJ + AEG + BEF	
CE + ABJ + ADG + BDF + FGJ	
CJ + DH + ABE + EFG	
DE + ABH + ACG + BCF + FGH	
EF + BCD + BHJ + CGJ + DGH	
EG + ACD + AHJ + CFJ + DFH	
EH + ABD + AGJ + BFJ + DFG	
EJ + ABC + AGH + BFH + CFG	
AEF + BEG + CEH + DEJ	

Factorial Plots for test -evaluation for segm ap

Bounding Box Minitab Regression Results:Bounding Box AP vs. 9 Factors

FFACTORS
Factorial Regression: test -evaluation ap for bbox versus LR, LRD, LRDecline, BATCH, ROI, IMPERCLASS, NUMCLASS, TYPE DSET,

Coded Coefficients

Term	Effect	Coef	SE Coef	95% CI	T-Value	P-Value	VIF
Constant		84.056	0.283	(83.498, 84.614)	297.53	0.000	
LR	-1.982	-0.991	0.283	(-1.549, -0.433)	-3.51	0.001	1.00
LRD	1.610	0.805	0.283	(0.247, 1.363)	2.85	0.005	1.00
LRDecline	0.135	0.067	0.283	(-0.491, 0.626)	0.24	0.812	1.00
BATCH	0.797	0.399	0.283	(-0.160, 0.957)	1.41	0.160	1.00
ROI	3.367	1.684	0.283	(1.125, 2.242)	5.96	0.000	1.00
IMPERCLASS	0.903	0.452	0.283	(-0.107, 1.010)	1.60	0.112	1.00
NUMCLASS	-5.071	-2.535	0.283	(-3.094, -1.977)	-8.97	0.000	1.00
TYPE DSET	1.447	0.723	0.283	(0.165, 1.282)	2.56	0.011	1.00
EPOCH	3.039	1.519	0.283	(0.961, 2.078)	5.38	0.000	1.00

Model Summary

S	R-sq	R-sq(adj)	PRESS	R-sq(pred)	AICc	BIC
3.57354	54.07%	51.31%	2179.44	47.74%	875.06	907.10

Analysis of Variance

Source	DF	Seq SS	Contribution	Adj SS	Adj MS	F-Value	P-Value
Model	9	2254.57	54.07%	2254.57	250.51	19.62	0.000
Linear	9	2254.57	54.07%	2254.57	250.51	19.62	0.000
LR	1	157.07	3.77%	157.07	157.07	12.30	0.001
LRD	1	103.63	2.49%	103.63	103.63	8.12	0.005
LRDecline	1	0.73	0.02%	0.73	0.73	0.06	0.812
BATCH	1	25.42	0.61%	25.42	25.42	1.99	0.160
ROI	1	453.48	10.87%	453.48	453.48	35.51	0.000
IMPERCLASS	1	32.62	0.78%	32.62	32.62	2.55	0.112
NUMCLASS	1	1028.56	24.67%	1028.56	1028.56	80.54	0.000
TYPE DSET	1	83.72	2.01%	83.72	83.72	6.56	0.011
EPOCH	1	369.33	8.86%	369.33	369.33	28.92	0.000
Error	150	1915.52	45.93%	1915.52	12.77		
Lack-of-Fit	22	1902.88	45.63%	1902.88	86.49	875.72	0.000
Pure Error	128	12.64	0.30%	12.64	0.10		
Total	159	4170.09	100.00%				

Regression Equation in Uncoded Units

test -evaluation ap for bbox = 83.10 - 82.6 LR + 17.88 LRD + 0.067 LRDecline + 0.199 BATCH
+ 0.00752 ROI + 0.00181 IMPERCLASS - 1.014 NUMCLASS
+ 0.723 TYPE DSET + 0.3039 EPOCH

Fits and Diagnostics for Unusual Observations

test -evaluation ap for bbox							
Obs	Fit	SE Fit	95% CI	Resid	Std Resid	Del Resid	HI
8	71.565	78.975	0.893 (77.210, 80.741)	-7.411	-2.14	-2.17	0.0625
34	73.877	80.855	0.893 (79.090, 82.621)	-6.978	-2.02	-2.04	0.0625
66	73.459	80.855	0.893 (79.090, 82.621)	-7.396	-2.14	-2.16	0.0625
104	71.519	78.975	0.893 (77.210, 80.741)	-7.456	-2.15	-2.18	0.0625
130	73.776	80.855	0.893 (79.090, 82.621)	-7.079	-2.05	-2.07	0.0625
Obs	Cook's D	DFITS					
8	0.03	-0.559798	R				
34	0.03	-0.526170	R				
66	0.03	-0.558660	R				
104	0.03	-0.563332	R				
130	0.03	-0.534002	R				

R Large residual

Alias Structure (up to order 3)

Factor	Name
A	LR
B	LRD
C	LRDecline
D	BATCH
E	ROI
F	IMPERCLASS
G	NUMCLASS
H	TYPE DSET
J	EPOCH
Aliases	
I	
A + BFG + CFH + DFJ	
B + AFG + CGH + DGJ	

C + AFH + BGH + DHJ
D + AFJ + BGJ + CHJ
E
F + ABG + ACH + ADJ
G + ABF + BCH + BDJ
H + ACF + BCG + CDJ
J + ADF + BDG + CDH

Segmentation Minitab Regression Results: Segmentation AP vs. 9 Factors

FFACTORS

Factorial Regression: test -evaluation for segm ap versus LR, LRD, LRDecline, BATCH, ROI, IMPERCLASS, NUMCLASS, TYPE DSET

Coded Coefficients

Term	Effect	Coef	SE Coef	95% CI	T-Value	P-Value	VIF
Constant		83.552	0.254	(83.049, 84.055)	328.36	0.000	
LR	-0.095	-0.047	0.254	(-0.550, 0.455)	-0.19	0.853	1.00
LRD	0.951	0.475	0.254	(-0.027, 0.978)	1.87	0.064	1.00
LRDecline	0.180	0.090	0.254	(-0.413, 0.593)	0.35	0.725	1.00
BATCH	0.210	0.105	0.254	(-0.398, 0.608)	0.41	0.681	1.00
ROI	1.965	0.982	0.254	(0.480, 1.485)	3.86	0.000	1.00
IMPERCLASS	0.737	0.369	0.254	(-0.134, 0.871)	1.45	0.150	1.00
NUMCLASS	-4.558	-2.279	0.254	(-2.782, -1.776)	-8.96	0.000	1.00
TYPE DSET	3.321	1.660	0.254	(1.158, 2.163)	6.53	0.000	1.00
EPOCH	2.230	1.115	0.254	(0.612, 1.618)	4.38	0.000	1.00

Model Summary

S	R-sq	R-sq(adj)	PRESS	R-sq(pred)	AICc	BIC
3.21860	52.05%	49.17%	1768.01	45.44%	841.58	873.62

Analysis of Variance

Source	DF	Seq SS	Contribution	Adj SS	Adj MS	F-Value	P-Value
Model	9	1686.73	52.05%	1686.73	187.415	18.09	0.000
Linear	9	1686.73	52.05%	1686.73	187.415	18.09	0.000
LR	1	0.36	0.01%	0.36	0.358	0.03	0.853
LRD	1	36.15	1.12%	36.15	36.153	3.49	0.064
LRDecline	1	1.29	0.04%	1.29	1.290	0.12	0.725
BATCH	1	1.76	0.05%	1.76	1.760	0.17	0.681
ROI	1	154.39	4.76%	154.39	154.392	14.90	0.000
IMPERCLASS	1	21.74	0.67%	21.74	21.738	2.10	0.150
NUMCLASS	1	830.93	25.64%	830.93	830.931	80.21	0.000
TYPE DSET	1	441.11	13.61%	441.11	441.114	42.58	0.000
EPOCH	1	199.00	6.14%	199.00	198.997	19.21	0.000
Error	150	1553.91	47.95%	1553.91	10.359		
Lack-of-Fit	22	1542.62	47.60%	1542.62	70.119	795.21	0.000
Pure Error	128	11.29	0.35%	11.29	0.088		
Total	159	3240.64	100.00%				

Regression Equation in Uncoded Units

test -evaluation for segm ap = 84.21 - 3.9 LR + 10.56 LRD + 0.090 LRDecline + 0.052 BATCH + 0.00439 ROI + 0.00147 IMPERCLASS - 0.912 NUMCLASS + 1.660 TYPE DSET + 0.2230 EPOCH

Alias Structure (up to order 3)

Factor	Name
A	LR
B	LRD
C	LRDecline
D	BATCH
E	ROI
F	IMPERCLASS
G	NUMCLASS
H	TYPE DSET
J	EPOCH
Aliases	
I	A + BFG + CFH + DFJ
B	A + AFG + CGH + DGJ
C	A + AFH + BGH + DHJ
D	A + AFJ + BGJ + CHJ
E	F + ABG + ACH + ADJ
G	A + ABF + BCH + BDJ
H	A + ACF + BCG + CDJ
J	A + ADF + BDG + CDH

Training Speed Minitab Regression Results

CORRECT LABEL FULL DESING

Factorial Regression: training speed (s/it) versus LR, LRD, LR Decline, Batch, ROI, NumImagesperClass, NumClasses, Dataset, epc

Coded Coefficients

Term	Effect	Coef	SE Coef	T-Value	P-Value	VIF
Constant		0.337990	0.000094	3596.38	0.000	
LR	0.005181	0.002590	0.000094	27.56	0.000	1.00
LRD	0.000548	0.000274	0.000094	2.92	0.004	1.00
LR Decline	-0.000072	-0.000036	0.000094	-0.38	0.704	1.00
Batch	0.212522	0.106261	0.000094	1130.67	0.000	1.00
ROI	0.091517	0.045758	0.000094	486.89	0.000	1.00
NumImagesperClass	-0.001171	-0.000585	0.000094	-6.23	0.000	1.00
NumClasses	0.009392	0.004696	0.000094	49.97	0.000	1.00
Dataset	0.006062	0.003031	0.000094	32.25	0.000	1.00
epoch	0.002153	0.001077	0.000094	11.46	0.000	1.00
LR*LRD	0.001887	0.000943	0.000094	10.04	0.000	1.00
LR*LR Decline	0.002447	0.001223	0.000094	13.02	0.000	1.00
LR*Batch	0.002738	0.001369	0.000094	14.57	0.000	1.00
LR*ROI	0.005213	0.002607	0.000094	27.74	0.000	1.00
LR*NumImagesperClass	0.000471	0.000235	0.000094	2.50	0.014	1.00
LR*NumClasses	-0.000887	-0.000493	0.000094	-5.25	0.000	1.00
LR*Dataset	0.001913	0.000957	0.000094	10.18	0.000	1.00
LR*epoch	-0.005483	-0.002742	0.000094	-29.17	0.000	1.00
LRD*LR Decline	-0.004706	-0.002353	0.000094	-25.04	0.000	1.00
LRD*Batch	0.001246	0.000623	0.000094	6.63	0.000	1.00
LRD*ROI	0.001111	0.000555	0.000094	5.91	0.000	1.00
LRD*Dataset	0.002836	0.001418	0.000094	15.09	0.000	1.00
LRD*epoch	0.003279	0.001640	0.000094	17.45	0.000	1.00
LR Decline*Batch	-0.000549	-0.000275	0.000094	-2.92	0.004	1.00
LR Decline*ROI	-0.000464	-0.000232	0.000094	-2.47	0.015	1.00
LR Decline*epoch	0.001744	0.000872	0.000094	9.28	0.000	1.00
Batch*ROI	0.028954	0.014477	0.000094	154.04	0.000	1.00
ROI*NumImagesperClass	-0.001318	-0.000659	0.000094	-7.01	0.000	1.00
ROI*NumClasses	0.009034	0.004517	0.000094	48.06	0.000	1.00
ROI*Dataset	0.006809	0.003405	0.000094	36.23	0.000	1.00
ROI*epoch	0.002346	0.001173	0.000094	12.48	0.000	1.00
LR*ROI*NumImagesperClass	0.000688	0.000344	0.000094	3.66	0.000	1.00

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.0011888	99.99%	99.99%	99.99%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	31	2.19197	0.07071	50035.48	0.000
Linear	9	2.14795	0.23866	168883.43	0.000
LR	1	0.00107	0.00107	759.74	0.000
LRD	1	0.00001	0.00001	8.51	0.004
LR Decline	1	0.00000	0.00000	0.15	0.704
Batch	1	1.80662	1.80662	1278412.18	0.000
ROI	1	0.33501	0.33501	237063.59	0.000
NumImagesperClass	1	0.00005	0.00005	38.80	0.000
NumClasses	1	0.00353	0.00353	2496.60	0.000
Dataset	1	0.00147	0.00147	1040.04	0.000
epoch	1	0.00019	0.00019	131.25	0.000
2-Way Interactions	21	0.04400	0.00210	1482.64	0.000
LR*LRD	1	0.00014	0.00014	100.75	0.000
LR*LR Decline	1	0.00024	0.00024	169.44	0.000
LR*Batch	1	0.00030	0.00030	212.24	0.000
LR*ROI	1	0.00109	0.00109	769.30	0.000
LR*NumImagesperClass	1	0.00001	0.00001	6.27	0.014
LR*NumClasses	1	0.00004	0.00004	27.56	0.000
LR*Dataset	1	0.00015	0.00015	103.62	0.000
LR*epoch	1	0.00120	0.00120	851.05	0.000
LRD*LR Decline	1	0.00089	0.00089	626.81	0.000
LRD*Batch	1	0.00006	0.00006	43.93	0.000
LRD*ROI	1	0.00005	0.00005	34.93	0.000
LRD*Dataset	1	0.00032	0.00032	227.63	0.000
LRD*epoch	1	0.00043	0.00043	304.36	0.000
LR Decline*Batch	1	0.00001	0.00001	8.54	0.004
LR Decline*ROI	1	0.00001	0.00001	6.10	0.015
LR Decline*epoch	1	0.00012	0.00012	86.11	0.000
Batch*ROI	1	0.03353	0.03353	23729.39	0.000
ROI*NumImagesperClass	1	0.00007	0.00007	49.19	0.000
ROI*NumClasses	1	0.00326	0.00326	2310.15	0.000
ROI*Dataset	1	0.00185	0.00185	1312.36	0.000
ROI*epoch	1	0.00022	0.00022	155.76	0.000
3-Way Interactions	1	0.00002	0.00002	13.41	0.000
LR*ROI*NumImagesperClass	1	0.00002	0.00002	13.41	0.000
Error	128	0.00018	0.00000		

Total 159 2.19215

Regression Equation in Uncoded Units

training speed (s/rt) = 0.02379 + 0.1648 LR - 0.14688 LRD + 0.000893 LR Decline
+ 0.042701 Batch - 0.000053 ROI + 0.000002 NumImagesperClass
- 0.000231 NumClasses - 0.004116 Dataset + 0.000107 epoch
+ 1.747 LR*LRD + 0.10194 LR*LR Decline + 0.05705 LR*Batch
+ 0.000330 LR*ROI - 0.000069 LR*NumImagesperClass
- 0.01644 LR*NumClasses + 0.07972 LR*Dataset - 0.04569 LR*epoch
- 0.05229 LRD*LR Decline + 0.00692 LRD*Batch + 0.000055 LRD*ROI
+ 0.03151 LRD*Dataset + 0.007287 LRD*epoch
- 0.000137 LR Decline*Batch - 0.000001 LR Decline*ROI
+ 0.000174 LR Decline*epoch + 0.000032 Batch*ROI
- 0.000000 ROI*NumImagesperClass + 0.000008 ROI*NumClasses
+ 0.000015 ROI*Dataset + 0.000001 ROI*epoch
+ 0.000001 LR*ROI*NumImagesperClass

Fits and Diagnostics for Unusual Observations

training				
Obs	speed (s/rt)	Fit	Resid	Std Resid
51	0.260100	0.263140	-0.003040	-2.86 R
52	0.270400	0.268080	0.002320	2.18 R
61	0.503900	0.498400	0.005500	5.17 R
91	0.510100	0.512940	-0.002840	-2.67 R
92	0.513200	0.515600	-0.002400	-2.26 R
94	0.514100	0.516360	-0.002260	-2.13 R
127	0.470700	0.468380	0.002320	2.18 R
147	0.266100	0.263140	0.002960	2.78 R
156	0.518900	0.515600	0.003300	3.10 R
157	0.495200	0.498400	-0.003200	-3.01 R

R Large residual

Alias Structure (up to order 3)

Factor	Name
A	LR
B	LRD
C	LR Decline
D	Batch
E	ROI
F	NumImagesperClass
G	NumClasses
H	Dataset
J	epoch
Aliases	
I	
A + BFG + CFH + DFJ	
B + AFG + CGH + DGJ	
C + AFH + BGH + DHJ	
D + AFJ + BGJ + CHJ	
E	
F + ABG + ACH + ADJ	
G + ABF + BCH + BDJ	
H + ACF + BCG + CDJ	
J + ADF + BDG + CDH	
AB + FG + CEJ + DEH	
AC + FH + BEJ + DEG	
AD + FJ + BEH + CEG	
AE + BCJ + BDH + CDG + GHJ	
AF + BG + CH + DJ	
AG + BF + CDE + EHJ	
AH + CF + BDE + EGJ	
AJ + DF + BCE + EGH	
BC + GH + AEJ + DEF	
BD + GJ + AEH + CEF	
BE + ACJ + ADH + CDF + FHJ	
BH + CG + ADE + EFJ	
BJ + DG + ACE + EFH	
CD + HJ + AEG + BEF	
CE + ABJ + ADG + BDF + FGJ	
CJ + DH + ABE + EFG	
DE + ABH + ACG + BCF + FGH	
EF + BCD + BHJ + CGJ + DGH	
EG + ACD + AHJ + CFJ + DFH	
EH + ABD + AGJ + BFJ + DFG	
EJ + ABC + AGH + BFH + CFG	
AEF + BEG + CEH + DEJ

APPENDIX G

General Linear Models

Iteration Factors Bounding Box Minitab General Linear Model: Bounding Box

AP vs. Number of Classes, Images per Class, Batch and Epoch

INTERACTITERS

General Linear Model: test -evaluation ap for bbox versus NUMCLASS, IMPERCLASS, BATCH, EPOCH

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
NUMCLASS	Fixed	2	5, 10
IMPERCLASS	Fixed	2	1000, 1500
BATCH	Fixed	2	4, 8
EPOCH	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
NUMCLASS	1	1028.56	1028.56	67.85	0.000
IMPERCLASS	1	32.62	32.62	2.15	0.145
BATCH	1	25.42	25.42	1.68	0.197
EPOCH	1	369.33	369.33	24.36	0.000
NUMCLASS*IMPERCLASS	1	39.48	39.48	2.60	0.109
NUMCLASS*BATCH	1	2.17	2.17	0.14	0.706
NUMCLASS*EPOCH	1	12.12	12.12	0.80	0.373
IMPERCLASS*BATCH	1	52.78	52.78	3.48	0.064
IMPERCLASS*EPOCH	1	133.82	133.82	8.83	0.003
BATCH*EPOCH	1	6.08	6.08	0.40	0.527
NUMCLASS*IMPERCLASS*BATCH	1	3.69	3.69	0.24	0.623
NUMCLASS*IMPERCLASS*EPOCH	1	5.08	5.08	0.33	0.564
NUMCLASS*BATCH*EPOCH	1	103.63	103.63	6.84	0.010
IMPERCLASS*BATCH*EPOCH	1	157.07	157.07	10.36	0.002
Error	145	2198.23	15.16		
Lack-of-Fit	1	3.19	3.19	0.21	0.648
Pure Error	144	2195.04	15.24		
Total	159	4170.09			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
3.89361	47.29%	42.20%	35.82%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	84.056	0.308	273.07	0.000	
NUMCLASS					
5	2.535	0.308	8.24	0.000	1.00
IMPERCLASS					
1000	-0.452	0.308	-1.47	0.145	1.00
BATCH					
4	-0.399	0.308	-1.29	0.197	1.00
EPOCH					
5	-1.519	0.308	-4.94	0.000	1.00
NUMCLASS*IMPERCLASS					
5 1000	0.497	0.308	1.61	0.109	1.00
NUMCLASS*BATCH					
5 4	-0.117	0.308	-0.38	0.706	1.00
NUMCLASS*EPOCH					
5 5	0.275	0.308	0.89	0.373	1.00
IMPERCLASS*BATCH					
1000 4	0.574	0.308	1.87	0.064	1.00
IMPERCLASS*EPOCH					
1000 5	0.915	0.308	2.97	0.003	1.00
BATCH*EPOCH					
4 5	0.195	0.308	0.63	0.527	1.00
NUMCLASS*IMPERCLASS*BATCH					
5 1000 4	0.152	0.308	0.49	0.623	1.00
NUMCLASS*IMPERCLASS*EPOCH					

5 1000 5	-0.178	0.308	-0.58	0.564	1.00
NUMCLASS*BATCH*EPOCH					
5 4 5	-0.805	0.308	-2.61	0.010	1.00
IMPERCLASS*BATCH*EPOCH					
1000 4 5	0.991	0.308	3.22	0.002	1.00

Regression Equation

test -evaluation ap for bbox = 84.056 + 2.535 NUMCLASS_5 - 2.535 NUMCLASS_10
- 0.452 IMPERCLASS_1000 + 0.452 IMPERCLASS_1500
- 0.399 BATCH_4 + 0.399 BATCH_8 - 1.519 EPOCH_5
+ 1.519 EPOCH_15 + 0.497 NUMCLASS*IMPERCLASS_5 1000
- 0.497 NUMCLASS*IMPERCLASS_5 1500
- 0.497 NUMCLASS*IMPERCLASS_10 1000
+ 0.497 NUMCLASS*IMPERCLASS_10 1500 - 0.117 NUMCLASS*BATCH_5 4
+ 0.117 NUMCLASS*BATCH_5 8 + 0.117 NUMCLASS*BATCH_10 4
- 0.117 NUMCLASS*BATCH_10 8 + 0.275 NUMCLASS*EPOCH_5 5
- 0.275 NUMCLASS*EPOCH_5 15 - 0.275 NUMCLASS*EPOCH_10 5
+ 0.275 NUMCLASS*EPOCH_10 15 + 0.574 IMPERCLASS*BATCH_1000 4
- 0.574 IMPERCLASS*BATCH_1000 8 - 0.574 IMPERCLASS*BATCH_1500
4 + 0.574 IMPERCLASS*BATCH_1500 8
+ 0.915 IMPERCLASS*EPOCH_1000 5 - 0.915 IMPERCLASS*EPOCH_1000
15 - 0.915 IMPERCLASS*EPOCH_1500 5
+ 0.915 IMPERCLASS*EPOCH_1500 15 + 0.195 BATCH*EPOCH_4 5
- 0.195 BATCH*EPOCH_4 15 - 0.195 BATCH*EPOCH_8 5
+ 0.195 BATCH*EPOCH_8 15 + 0.152 NUMCLASS*IMPERCLASS*BATCH_5
1000 4 - 0.152 NUMCLASS*IMPERCLASS*BATCH_5 1000 8
- 0.152 NUMCLASS*IMPERCLASS*BATCH_5 1500 4
+ 0.152 NUMCLASS*IMPERCLASS*BATCH_5 1500 8
- 0.152 NUMCLASS*IMPERCLASS*BATCH_10 1000 4
+ 0.152 NUMCLASS*IMPERCLASS*BATCH_10 1000 8
+ 0.152 NUMCLASS*IMPERCLASS*BATCH_10 1500 4
- 0.152 NUMCLASS*IMPERCLASS*BATCH_10 1500 8
- 0.178 NUMCLASS*IMPERCLASS*EPOCH_5 1000 5
+ 0.178 NUMCLASS*IMPERCLASS*EPOCH_5 1000 15
+ 0.178 NUMCLASS*IMPERCLASS*EPOCH_5 1500 5
- 0.178 NUMCLASS*IMPERCLASS*EPOCH_5 1500 15
+ 0.178 NUMCLASS*IMPERCLASS*EPOCH_10 1000 5
- 0.178 NUMCLASS*IMPERCLASS*EPOCH_10 1000 15
- 0.178 NUMCLASS*IMPERCLASS*EPOCH_10 1500 5
+ 0.178 NUMCLASS*IMPERCLASS*EPOCH_10 1500 15
- 0.805 NUMCLASS*BATCH*EPOCH_5 4 5
+ 0.805 NUMCLASS*BATCH*EPOCH_5 4 15
+ 0.805 NUMCLASS*BATCH*EPOCH_5 8 5
- 0.805 NUMCLASS*BATCH*EPOCH_5 8 15
+ 0.805 NUMCLASS*BATCH*EPOCH_10 4 5
- 0.805 NUMCLASS*BATCH*EPOCH_10 4 15
- 0.805 NUMCLASS*BATCH*EPOCH_10 8 5
+ 0.805 NUMCLASS*BATCH*EPOCH_10 8 15
+ 0.991 IMPERCLASS*BATCH*EPOCH_1000 4 5
- 0.991 IMPERCLASS*BATCH*EPOCH_1000 4 15
- 0.991 IMPERCLASS*BATCH*EPOCH_1000 8 5
+ 0.991 IMPERCLASS*BATCH*EPOCH_1000 8 15
- 0.991 IMPERCLASS*BATCH*EPOCH_1500 4 5
+ 0.991 IMPERCLASS*BATCH*EPOCH_1500 4 15
+ 0.991 IMPERCLASS*BATCH*EPOCH_1500 8 5
- 0.991 IMPERCLASS*BATCH*EPOCH_1500 8 15

Fits and Diagnostics for Unusual Observations

		test			
		-evaluation			
Obs	ap for bbox	Fit	Resid	Std Resid	
2	73.95	81.72	-7.77	-2.10	R
22	89.48	81.72	7.75	2.09	R
34	73.88	81.72	-7.85	-2.12	R
54	90.12	81.72	8.40	2.27	R
66	73.46	81.72	-8.26	-2.23	R
86	89.98	81.72	8.25	2.23	R
118	89.69	81.72	7.96	2.15	R
130	73.78	81.72	-7.95	-2.14	R
150	89.79	81.72	8.06	2.17	R

R Large residual

Factorial Plots for test -evaluation ap for bbox

Iteration Factors Segmentation Minitab General Linear Model: Segmentation AP vs. Number of Classes, Images per Class, Batch and Epoch

INTERACTITERS

General Linear Model: test -evaluation for segm ap versus NUMCLASS, IMPERCLASS, BATCH, EPOCH

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
NUMCLASS	Fixed	2	5, 10
IMPERCLASS	Fixed	2	1000, 1500
BATCH	Fixed	2	4, 8
EPOCH	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
NUMCLASS	1	830.93	830.931	58.88	0.000
IMPERCLASS	1	21.74	21.738	1.54	0.217
BATCH	1	1.76	1.760	0.12	0.724
EPOCH	1	199.00	198.997	14.10	0.000
NUMCLASS*IMPERCLASS	1	8.29	8.292	0.59	0.445
NUMCLASS*BATCH	1	0.97	0.972	0.07	0.793
NUMCLASS*EPOCH	1	19.22	19.217	1.36	0.245
IMPERCLASS*BATCH	1	11.38	11.377	0.81	0.371
IMPERCLASS*EPOCH	1	50.93	50.928	3.61	0.059
BATCH*EPOCH	1	7.38	7.376	0.52	0.471
NUMCLASS*IMPERCLASS*BATCH	1	2.12	2.123	0.15	0.699
NUMCLASS*IMPERCLASS*EPOCH	1	4.09	4.092	0.29	0.591
NUMCLASS*BATCH*EPOCH	1	36.15	36.153	2.56	0.112
IMPERCLASS*BATCH*EPOCH	1	0.36	0.358	0.03	0.874
Error	145	2046.33	14.113		
Lack-of-Fit	1	0.28	0.277	0.02	0.889
Pure Error	144	2046.05	14.209		
Total	159	3240.64			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
3.75668	36.85%	30.76%	23.11%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	83.552	0.297	281.33	0.000	
NUMCLASS					
5	2.279	0.297	7.67	0.000	1.00
IMPERCLASS					
1000	-0.369	0.297	-1.24	0.217	1.00
BATCH					
4	-0.105	0.297	-0.35	0.724	1.00
EPOCH					
5	-1.115	0.297	-3.76	0.000	1.00
NUMCLASS*IMPERCLASS					
5 1000	0.228	0.297	0.77	0.445	1.00
NUMCLASS*BATCH					
5 4	-0.078	0.297	-0.26	0.793	1.00
NUMCLASS*EPOCH					
5 5	0.347	0.297	1.17	0.245	1.00
IMPERCLASS*BATCH					
1000 4	0.267	0.297	0.90	0.371	1.00
IMPERCLASS*EPOCH					
1000 5	0.564	0.297	1.90	0.059	1.00
BATCH*EPOCH					
4 5	0.215	0.297	0.72	0.471	1.00
NUMCLASS*IMPERCLASS*BATCH					
5 1000 4	0.115	0.297	0.39	0.699	1.00
NUMCLASS*IMPERCLASS*EPOCH					

5 1000 5	-0.160	0.297	-0.54	0.591	1.00
NUMCLASS*BATCH*EPOCH					
5 4 5	-0.475	0.297	-1.60	0.112	1.00
IMPERCLASS*BATCH*EPOCH					
1000 4 5	0.047	0.297	0.16	0.874	1.00

Regression Equation

```
test -evaluation for segm ap = 83.552 + 2.279 NUMCLASS_5 - 2.279 NUMCLASS_10
- 0.369 IMPERCLASS_1000 + 0.369 IMPERCLASS_1500
- 0.105 BATCH_4 + 0.105 BATCH_8 - 1.115 EPOCH_5
+ 1.115 EPOCH_15 + 0.228 NUMCLASS*IMPERCLASS_5 1000
- 0.228 NUMCLASS*IMPERCLASS_5 1500
- 0.228 NUMCLASS*IMPERCLASS_10 1000
+ 0.228 NUMCLASS*IMPERCLASS_10 1500 - 0.078 NUMCLASS*BATCH_5 4
+ 0.078 NUMCLASS*BATCH_5 8 + 0.078 NUMCLASS*BATCH_10 4
- 0.078 NUMCLASS*BATCH_10 8 + 0.347 NUMCLASS*EPOCH_5 5
- 0.347 NUMCLASS*EPOCH_5 15 - 0.347 NUMCLASS*EPOCH_10 5
+ 0.347 NUMCLASS*EPOCH_10 15 + 0.267 IMPERCLASS*BATCH_1000 4
- 0.267 IMPERCLASS*BATCH_1000 8 - 0.267 IMPERCLASS*BATCH_1500
4 + 0.267 IMPERCLASS*BATCH_1500 8
+ 0.564 IMPERCLASS*EPOCH_1000 5 - 0.564 IMPERCLASS*EPOCH_1000
15 - 0.564 IMPERCLASS*EPOCH_1500 5
+ 0.564 IMPERCLASS*EPOCH_1500 15 + 0.215 BATCH*EPOCH_4 5
- 0.215 BATCH*EPOCH_4 15 - 0.215 BATCH*EPOCH_8 5
+ 0.215 BATCH*EPOCH_8 15 + 0.115 NUMCLASS*IMPERCLASS*BATCH_5
1000 4 - 0.115 NUMCLASS*IMPERCLASS*BATCH_5 1000 8
- 0.115 NUMCLASS*IMPERCLASS*BATCH_5 1500 4
+ 0.115 NUMCLASS*IMPERCLASS*BATCH_5 1500 8
- 0.115 NUMCLASS*IMPERCLASS*BATCH_10 1000 4
+ 0.115 NUMCLASS*IMPERCLASS*BATCH_10 1000 8
- 0.115 NUMCLASS*IMPERCLASS*BATCH_10 1500 4
- 0.115 NUMCLASS*IMPERCLASS*BATCH_10 1500 8
- 0.160 NUMCLASS*IMPERCLASS*EPOCH_5 1000 5
+ 0.160 NUMCLASS*IMPERCLASS*EPOCH_5 1000 15
+ 0.160 NUMCLASS*IMPERCLASS*EPOCH_5 1500 5
- 0.160 NUMCLASS*IMPERCLASS*EPOCH_5 1500 15
+ 0.160 NUMCLASS*IMPERCLASS*EPOCH_10 1000 5
- 0.160 NUMCLASS*IMPERCLASS*EPOCH_10 1000 15
- 0.160 NUMCLASS*IMPERCLASS*EPOCH_10 1500 5
+ 0.160 NUMCLASS*IMPERCLASS*EPOCH_10 1500 15
- 0.475 NUMCLASS*BATCH*EPOCH_5 4 5
+ 0.475 NUMCLASS*BATCH*EPOCH_5 4 15
+ 0.475 NUMCLASS*BATCH*EPOCH_5 8 5
- 0.475 NUMCLASS*BATCH*EPOCH_5 8 15
+ 0.475 NUMCLASS*BATCH*EPOCH_10 4 5
- 0.475 NUMCLASS*BATCH*EPOCH_10 4 15
- 0.475 NUMCLASS*BATCH*EPOCH_10 8 5
+ 0.475 NUMCLASS*BATCH*EPOCH_10 8 15
+ 0.047 IMPERCLASS*BATCH*EPOCH_1000 4 5
- 0.047 IMPERCLASS*BATCH*EPOCH_1000 4 15
- 0.047 IMPERCLASS*BATCH*EPOCH_1000 8 5
+ 0.047 IMPERCLASS*BATCH*EPOCH_1000 8 15
- 0.047 IMPERCLASS*BATCH*EPOCH_1500 4 5
+ 0.047 IMPERCLASS*BATCH*EPOCH_1500 4 15
+ 0.047 IMPERCLASS*BATCH*EPOCH_1500 8 5
- 0.047 IMPERCLASS*BATCH*EPOCH_1500 8 15
```

Fits and Diagnostics for Unusual Observations

test		-evaluation			
Obs	for segm ap	Fit	Resid	Std Resid	
66	76.72	83.93	-7.21	-2.02	R

R Large residual

Factorial Plots for test -evaluation for segm ap

Hyperparameters General Linear Model of the Bounding Box vs Learning Rate, Learning Rate Decline, Batch, ROI Epochs and Interactions

MODEL PARAMETERS

General Linear Model: test -evaluation ap for bbox versus LR, LRD, LRDdecline, Batch, ROI, Epochs

The following terms cannot be estimated and were removed:
LR*LRD*LRDdecline, LR*LRD*ROI, LR*LRD*Epochs, LR*LRDdecline*ROI, LR*LRDdecline*Epochs, LR*Batch*Epochs, LR*ROI*Epochs, LRD*LRDdecline*Batch, LRD*LRDdecline*ROI, LRD*LRDdecline*Epochs, LRD*ROI*Epochs, LRDdecline*Batch*ROI, LRDdecline*Batch*Epochs, LRDdecline*ROI*Epochs, Batch*ROI*Epochs, LR*LRD*LRDdecline*Batch, LR*LRD*LRDdecline*ROI, LR*LRD*LRDdecline*Epochs, LR*LRD*Batch*ROI, LR*LRD*Batch*Epochs, LR*LRD*ROI*Epochs, LR*LRDdecline*Batch*ROI, LR*LRDdecline*Batch*Epochs, LR*LRDdecline*ROI*Epochs, LR*Batch*ROI*Epochs, LRD*LRDdecline*Batch*ROI, LRD*LRDdecline*Batch*Epochs, LRD*LRDdecline*ROI*Epochs, LRD*Batch*ROI*Epochs, LRDdecline*Batch*ROI*Epochs, LR*LRD*LRDdecline*Batch*ROI, LR*LRD*LRDdecline*Batch*Epochs, LR*LRD*LRDdecline*ROI*Epochs, LR*LRD*Batch*ROI*Epochs, LR*LRDdecline*Batch*ROI*Epochs, LRD*LRDdecline*Batch*ROI*Epochs, LR*LRD*LRDdecline*Batch*ROI*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDdecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	57.75	57.748	759.07	0.000
LRD	1	2.89	2.886	37.93	0.000
LRDdecline	1	30.32	30.324	398.60	0.000
Batch	1	16.56	16.559	217.66	0.000
ROI	1	204.91	204.906	2693.38	0.000
Epochs	1	220.47	220.470	2897.97	0.000
LR*LRD	1	0.16	0.164	2.15	0.146
LR*LRDdecline	1	17.70	17.696	232.60	0.000
LR*Batch	1	46.37	46.369	609.50	0.000
LR*ROI	1	255.10	255.097	3353.12	0.000
LR*Epochs	1	73.73	73.730	969.15	0.000
LRD*LRDdecline	1	864.61	864.612	11364.90	0.000
LRD*Batch	1	48.72	48.719	640.39	0.000
LRD*ROI	1	14.06	14.059	184.80	0.000
LRD*Epochs	1	0.29	0.289	3.81	0.054
LRDdecline*Batch	1	61.76	61.755	811.74	0.000
LRDdecline*ROI	1	0.05	0.050	0.65	0.421
LRDdecline*Epochs	1	5.54	5.536	72.77	0.000
Batch*ROI	1	42.95	42.951	564.57	0.000
Batch*Epochs	1	23.50	23.501	308.90	0.000
ROI*Epochs	1	9.51	9.509	124.99	0.000
LR*LRD*Batch	1	23.40	23.395	307.52	0.000
LR*LRDdecline*Batch	1	35.78	35.775	470.25	0.000
LR*Batch*ROI	1	58.87	58.874	773.86	0.000
LRD*Batch*ROI	1	0.21	0.213	2.79	0.098
LRD*Batch*Epochs	1	849.58	849.584	11167.37	0.000
Error	104	7.91	0.076		
Total	130	4285.18			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.275821	99.82%	99.77%	*

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	83.3327	0.0308	2702.29	0.000	
LR					
0.001	0.8496	0.0308	27.55	0.000	1.64
LRD					
0.01	0.3290	0.0534	6.16	0.000	4.91
LRDdecline					
0.7	1.0664	0.0534	19.96	0.000	4.91
Batch					
4	-0.4550	0.0308	-14.75	0.000	1.64
ROI					
64	-2.7720	0.0534	-51.90	0.000	4.91
Epochs					
5	-2.3477	0.0436	-53.83	0.000	3.26
LR*LRD					
0.001 0.01	0.0452	0.0308	1.47	0.146	1.63
LR*LRDdecline					
0.001 0.7	-0.4703	0.0308	-15.25	0.000	1.63
LR*Batch					
0.001 4	0.7613	0.0308	24.69	0.000	1.63
LR*ROI					
0.001 64	2.5254	0.0436	57.91	0.000	3.26
LR*Epochs					
0.001 5	1.6628	0.0534	31.13	0.000	4.91
LRD*LRDdecline					
0.01 0.7	-3.2875	0.0308	-106.61	0.000	1.56
LRD*Batch					
0.01 4	1.1036	0.0436	25.31	0.000	3.26
LRD*ROI					
0.01 64	-0.4192	0.0308	-13.59	0.000	1.63
LRD*Epochs					
0.01 5	-0.0602	0.0308	-1.95	0.054	1.64
LRDdecline*Batch					
0.7 4	1.2425	0.0436	28.49	0.000	3.26
LRDdecline*ROI					
0.7 64	0.0249	0.0308	0.81	0.421	1.63
LRDdecline*Epochs					
0.7 5	-0.2631	0.0308	-8.53	0.000	1.64
Batch*ROI					
4 64	-0.7327	0.0308	-23.76	0.000	1.63
Batch*Epochs					
4 5	-0.9388	0.0534	-17.58	0.000	4.91
ROI*Epochs					
64 5	-0.3448	0.0308	-11.18	0.000	1.64
LR*LRD*Batch					
0.001 0.01 4	-0.9367	0.0534	-17.54	0.000	4.91
LR*LRDdecline*Batch					
0.001 0.7 4	-1.1583	0.0534	-21.69	0.000	4.91
LR*Batch*ROI					
0.001 4 64	1.4859	0.0534	27.82	0.000	4.91
LRD*Batch*ROI					
0.01 4 64	-0.0516	0.0308	-1.67	0.098	1.64
LRD*Batch*Epochs					
0.01 4 5	3.2588	0.0308	105.68	0.000	1.56

Regression Equation

test -evaluation ap for bbox = $83.3327 + 0.8496 LR_{0.001} - 0.8496 LR_{0.025} + 0.3290 LRD_{0.01} - 0.3290 LRD_{0.10} + 1.0664 LRDdecline_{0.7} - 1.0664 LRDdecline_{0.9} - 0.4550 Batch_{.4} + 0.4550 Batch_{.8} - 2.7720 ROI_{.64} + 2.7720 ROI_{.512} - 2.3477 Epochs_{.5} + 2.3477 Epochs_{.15} + 0.0452 LR*LRD_{0.001\ 0.01} - 0.0452 LR*LRD_{0.001\ 0.10} - 0.0452 LR*LRD_{0.025\ 0.01} + 0.0452 LR*LRD_{0.025\ 0.10} - 0.4703 LR*LRDdecline_{0.001\ 0.7} + 0.4703 LR*LRDdecline_{0.001\ 0.9} + 0.4703 LR*LRDdecline_{0.025\ 0.7} - 0.4703 LR*LRDdecline_{0.025\ 0.9} + 0.7613 LR*Batch_{0.001\ 4} - 0.7613 LR*Batch_{0.001\ 8} - 0.7613 LR*Batch_{0.025\ 4} + 0.7613 LR*Batch_{0.025\ 8} + 2.5254 LR*ROI_{0.001\ 64} - 2.5254 LR*ROI_{0.001\ 512} - 2.5254 LR*ROI_{0.025\ 64} + 2.5254 LR*ROI_{0.025\ 512} + 1.6628 LR*Epochs_{0.001\ 5} - 1.6628 LR*Epochs_{0.001\ 15} - 1.6628 LR*Epochs_{0.025\ 5} + 1.6628 LR*Epochs_{0.025\ 15} - 3.2875 LRD*LRDdecline_{0.01\ 0.7}$

```

+ 3.2875 LRD*LRDdecline_0.01 0.9 + 3.2875 LRD*LRDdecline_0.10
0.7 - 3.2875 LRD*LRDdecline_0.10 0.9 + 1.1036 LRD*Batch_0.01 4
- 1.1036 LRD*Batch_0.01 8 - 1.1036 LRD*Batch_0.10 4
+ 1.1036 LRD*Batch_0.10 8 - 0.4192 LRD*ROI_0.01 64
+ 0.4192 LRD*ROI_0.01 512 + 0.4192 LRD*ROI_0.10 64
- 0.4192 LRD*ROI_0.10 512 - 0.0602 LRD*Epochs_0.01 5
+ 0.0602 LRD*Epochs_0.01 15 + 0.0602 LRD*Epochs_0.10 5
- 0.0602 LRD*Epochs_0.10 15 + 1.2425 LRDdecline*Batch_0.7 4
- 1.2425 LRDdecline*Batch_0.7 8 - 1.2425 LRDdecline*Batch_0.9
4 + 1.2425 LRDdecline*Batch_0.9 8 + 0.0249 LRDdecline*ROI_0.7
64 - 0.0249 LRDdecline*ROI_0.7 512 - 0.0249 LRDdecline*ROI_0.9
64 + 0.0249 LRDdecline*ROI_0.9 512
- 0.2631 LRDdecline*Epochs_0.7 5
+ 0.2631 LRDdecline*Epochs_0.7 15
+ 0.2631 LRDdecline*Epochs_0.9 5
- 0.2631 LRDdecline*Epochs_0.9 15 - 0.7327 Batch*ROI_4 64
+ 0.7327 Batch*ROI_4 512 + 0.7327 Batch*ROI_8 64
- 0.7327 Batch*ROI_8 512 - 0.9388 Batch*Epochs_4 5
+ 0.9388 Batch*Epochs_4 15 + 0.9388 Batch*Epochs_8 5
- 0.9388 Batch*Epochs_8 15 - 0.3448 ROI*Epochs_64 5
+ 0.3448 ROI*Epochs_64 15 + 0.3448 ROI*Epochs_512 5
- 0.3448 ROI*Epochs_512 15 - 0.9367 LR*LRD*Batch_0.001 0.01 4
+ 0.9367 LR*LRD*Batch_0.001 0.01 8 + 0.9367 LR*LRD*Batch_0.001
0.10 4 - 0.9367 LR*LRD*Batch_0.001 0.10 8
+ 0.9367 LR*LRD*Batch_0.025 0.01 4 - 0.9367 LR*LRD*Batch_0.025
0.01 8 - 0.9367 LR*LRD*Batch_0.025 0.10 4
+ 0.9367 LR*LRD*Batch_0.025 0.10 8
- 1.1583 LR*LRDdecline*Batch_0.001 0.7 4
+ 1.1583 LR*LRDdecline*Batch_0.001 0.7 8
+ 1.1583 LR*LRDdecline*Batch_0.001 0.9 4
- 1.1583 LR*LRDdecline*Batch_0.001 0.9 8
+ 1.1583 LR*LRDdecline*Batch_0.025 0.7 4
- 1.1583 LR*LRDdecline*Batch_0.025 0.7 8
- 1.1583 LR*LRDdecline*Batch_0.025 0.9 4
+ 1.1583 LR*LRDdecline*Batch_0.025 0.9 8
+ 1.4859 LR*Batch*ROI_0.001 4 64 - 1.4859 LR*Batch*ROI_0.001 4
512 - 1.4859 LR*Batch*ROI_0.001 8 64
+ 1.4859 LR*Batch*ROI_0.001 8 512 - 1.4859 LR*Batch*ROI_0.025
4 64 + 1.4859 LR*Batch*ROI_0.025 4 512
+ 1.4859 LR*Batch*ROI_0.025 8 64 - 1.4859 LR*Batch*ROI_0.025 8
512 - 0.0516 LRD*Batch*ROI_0.01 4 64
+ 0.0516 LRD*Batch*ROI_0.01 4 512 + 0.0516 LRD*Batch*ROI_0.01
8 64 - 0.0516 LRD*Batch*ROI_0.01 8 512
+ 0.0516 LRD*Batch*ROI_0.10 4 64 - 0.0516 LRD*Batch*ROI_0.10 4
512 - 0.0516 LRD*Batch*ROI_0.10 8 64
+ 0.0516 LRD*Batch*ROI_0.10 8 512
+ 3.2588 LRD*Batch*Epochs_0.01 4 5
- 3.2588 LRD*Batch*Epochs_0.01 4 15
- 3.2588 LRD*Batch*Epochs_0.01 8 5
+ 3.2588 LRD*Batch*Epochs_0.01 8 15
- 3.2588 LRD*Batch*Epochs_0.10 4 5
+ 3.2588 LRD*Batch*Epochs_0.10 4 15
+ 3.2588 LRD*Batch*Epochs_0.10 8 5
- 3.2588 LRD*Batch*Epochs_0.10 8 15

```

Fits and Diagnostics for Unusual Observations

test					
-evaluation					
Obs	ap for bbox	Fit	Resid	Std Resid	
5	74.538	73.921	0.617	2.50	R
18	84.210	83.638	0.572	2.32	R
59	77.281	76.762	0.520	2.11	R
122	67.879	67.371	0.508	2.06	R
125	66.568	67.371	-0.803	-3.26	R
131	75.721	75.721	0.000	*	X

R Large residual
X Unusual X

Hyperparameters General Linear Model of the Segmentation vs Learning Rate, Learning Rate Decline, Batch, ROI Epochs and Interactions

MODEL PARAMETERS

General Linear Model: test -evaluation for segm ap versus LR, LRD, LRDdecline, Batch, ROI, Epochs

The following terms cannot be estimated and were removed:

LR*LRD*LRDdecline, LR*LRD*ROI, LR*LRD*Epochs, LR*LRDdecline*ROI, LR*LRDdecline*Epochs, LR*Batch*Epochs, LR*ROI*Epochs, LRD*LRDdecline*Batch, LRD*LRDdecline*ROI, LRD*LRDdecline*Epochs, LRD*ROI*Epochs, LRDdecline*Batch*ROI, LRDdecline*Batch*Epochs, LRDdecline*ROI*Epochs, Batch*ROI*Epochs, LR*LRD*LRDdecline*Batch, LR*LRD*LRDdecline*ROI, LR*LRD*LRDdecline*Epochs, LR*LRD*Batch*ROI, LR*LRD*Batch*Epochs, LR*LRD*ROI*Epochs, LR*LRDdecline*Batch*ROI, LR*LRDdecline*Batch*Epochs, LR*LRDdecline*ROI*Epochs, LR*Batch*ROI*Epochs, LRD*LRDdecline*Batch*ROI, LRD*LRDdecline*Batch*Epochs, LRD*LRDdecline*ROI*Epochs, LRD*Batch*ROI*Epochs, LRDdecline*Batch*ROI*Epochs, LR*LRD*LRDdecline*Batch*ROI, LR*LRD*LRDdecline*Batch*Epochs, LR*LRD*LRDdecline*ROI*Epochs, LR*LRD*Batch*ROI*Epochs, LR*LRDdecline*Batch*ROI*Epochs, LRD*LRDdecline*Batch*ROI*Epochs, LR*LRD*LRDdecline*Batch*ROI*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDdecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	0.00	0.00	0.04	0.847
LRD	1	0.88	0.88	12.90	0.001
LRDdecline	1	1.10	1.10	16.12	0.000
Batch	1	1.67	1.67	24.35	0.000
ROI	1	40.41	40.41	589.78	0.000
Epochs	1	46.34	46.34	676.30	0.000
LR*LRD	1	1.59	1.59	23.19	0.000
LR*LRDdecline	1	11.67	11.67	170.34	0.000
LR*Batch	1	17.14	17.14	250.19	0.000
LR*ROI	1	40.81	40.81	595.56	0.000
LR*Epochs	1	7.08	7.08	103.37	0.000
LRD*LRDdecline	1	1559.13	1559.13	22754.01	0.000
LRD*Batch	1	3.79	3.79	55.26	0.000
LRD*ROI	1	3.18	3.18	46.40	0.000
LRD*Epochs	1	0.12	0.12	1.72	0.192
LRDdecline*Batch	1	5.07	5.07	73.99	0.000
LRDdecline*ROI	1	0.28	0.28	4.02	0.048
LRDdecline*Epochs	1	0.65	0.65	9.52	0.003
Batch*ROI	1	17.23	17.23	251.48	0.000
Batch*Epochs	1	0.16	0.16	2.40	0.124
ROI*Epochs	1	3.68	3.68	53.67	0.000
LR*LRD*Batch	1	0.48	0.48	6.94	0.010
LR*LRDdecline*Batch	1	1.35	1.35	19.77	0.000
LR*Batch*ROI	1	4.39	4.39	64.14	0.000
LRD*Batch*ROI	1	0.74	0.74	10.73	0.001
LRD*Batch*Epochs	1	1241.44	1241.44	18117.69	0.000
Error	104	7.13	0.07		
Total	130	3762.00			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.261765	99.81%	99.76%	*

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
------	------	---------	---------	---------	-----

Constant	81.8919	0.0293	2798.17	0.000	
LR					
0.001	0.0057	0.0293	0.19	0.847	1.64
LRD					
0.01	-0.1821	0.0507	-3.59	0.001	4.91
LRDdecline					
0.7	0.2035	0.0507	4.01	0.000	4.91
Batch					
4	-0.1444	0.0293	-4.93	0.000	1.64
ROI					
64	-1.2310	0.0507	-24.29	0.000	4.91
Epochs					
5	-1.0763	0.0414	-26.01	0.000	3.26
LR*LRD					
0.001 0.01	-0.1409	0.0293	-4.82	0.000	1.63
LR*LRDdecline					
0.001 0.7	-0.3820	0.0293	-13.05	0.000	1.63
LR*Batch					
0.001 4	0.4629	0.0293	15.82	0.000	1.63
LR*ROI					
0.001 64	1.0101	0.0414	24.40	0.000	3.26
LR*Epochs					
0.001 5	0.5154	0.0507	10.17	0.000	4.91
LRD*LRDdecline					
0.01 0.7	-4.4146	0.0293	-150.84	0.000	1.56
LRD*Batch					
0.01 4	0.3077	0.0414	7.43	0.000	3.26
LRD*ROI					
0.01 64	-0.1994	0.0293	-6.81	0.000	1.63
LRD*Epochs					
0.01 5	-0.0384	0.0293	-1.31	0.192	1.64
LRDdecline*Batch					
0.7 4	0.3560	0.0414	8.60	0.000	3.26
LRDdecline*ROI					
0.7 64	0.0587	0.0293	2.00	0.048	1.63
LRDdecline*Epochs					
0.7 5	-0.0903	0.0293	-3.09	0.003	1.64
Batch*ROI					
4 64	-0.4641	0.0293	-15.86	0.000	1.63
Batch*Epochs					
4 5	-0.0786	0.0507	-1.55	0.124	4.91
ROI*Epochs					
64 5	-0.2144	0.0293	-7.33	0.000	1.64
LR*LRD*Batch					
0.001 0.01 4	-0.1335	0.0507	-2.63	0.010	4.91
LR*LRDdecline*Batch					
0.001 0.7 4	-0.2254	0.0507	-4.45	0.000	4.91
LR*Batch*ROI					
0.001 4 64	0.4060	0.0507	8.01	0.000	4.91
LRD*Batch*ROI					
0.01 4 64	0.0959	0.0293	3.28	0.001	1.64
LRD*Batch*Epochs					
0.01 4 5	3.9393	0.0293	134.60	0.000	1.56

Regression Equation

```

test -evaluation for segm ap = 81.8919 + 0.0057 LR_0.001 - 0.0057 LR_0.025 - 0.1821 LRD_0.01
+ 0.1821 LRD_0.10 + 0.2035 LRDdecline_0.7
- 0.2035 LRDdecline_0.9 - 0.1444 Batch_4 + 0.1444 Batch_8
- 1.2310 ROI_64 + 1.2310 ROI_512 - 1.0763 Epochs_5
+ 1.0763 Epochs_15 - 0.1409 LR*LRD_0.001 0.01
+ 0.1409 LR*LRD_0.001 0.10 + 0.1409 LR*LRD_0.025 0.01
- 0.1409 LR*LRD_0.025 0.10 - 0.3820 LR*LRDdecline_0.001 0.7
+ 0.3820 LR*LRDdecline_0.001 0.9 + 0.3820 LR*LRDdecline_0.025
0.7 - 0.3820 LR*LRDdecline_0.025 0.9 + 0.4629 LR*Batch_0.001 4
- 0.4629 LR*Batch_0.001 8 - 0.4629 LR*Batch_0.025 4
+ 0.4629 LR*Batch_0.025 8 + 1.0101 LR*ROI_0.001 64
- 1.0101 LR*ROI_0.001 512 - 1.0101 LR*ROI_0.025 64
+ 1.0101 LR*ROI_0.025 512 + 0.5154 LR*Epochs_0.001 5
- 0.5154 LR*Epochs_0.001 15 - 0.5154 LR*Epochs_0.025 5
+ 0.5154 LR*Epochs_0.025 15 - 4.4146 LRD*LRDdecline_0.01 0.7
+ 4.4146 LRD*LRDdecline_0.01 0.9 + 4.4146 LRD*LRDdecline_0.10

```

```

0.7 - 4.4146 LRD*LRDdecline_0.10 0.9 + 0.3077 LRD*Batch_0.01 4
- 0.3077 LRD*Batch_0.01 8 - 0.3077 LRD*Batch_0.10 4
+ 0.3077 LRD*Batch_0.10 8 - 0.1994 LRD*ROI_0.01 64
+ 0.1994 LRD*ROI_0.01 512 + 0.1994 LRD*ROI_0.10 64
- 0.1994 LRD*ROI_0.10 512 - 0.0384 LRD*Epochs_0.01 5
+ 0.0384 LRD*Epochs_0.01 15 + 0.0384 LRD*Epochs_0.10 5
- 0.0384 LRD*Epochs_0.10 15 + 0.3560 LRDdecline*Batch_0.7 4
- 0.3560 LRDdecline*Batch_0.7 8 - 0.3560 LRDdecline*Batch_0.9
4 + 0.3560 LRDdecline*Batch_0.9 8 + 0.0587 LRDdecline*ROI_0.7
64 - 0.0587 LRDdecline*ROI_0.7 512 - 0.0587 LRDdecline*ROI_0.9
64 + 0.0587 LRDdecline*ROI_0.9 512
- 0.0903 LRDdecline*Epochs_0.7 5
+ 0.0903 LRDdecline*Epochs_0.7 15
+ 0.0903 LRDdecline*Epochs_0.9 5
- 0.0903 LRDdecline*Epochs_0.9 15 - 0.4641 Batch*ROI_4 64
+ 0.4641 Batch*ROI_4 512 + 0.4641 Batch*ROI_8 64
- 0.4641 Batch*ROI_8 512 - 0.0786 Batch*Epochs_4 5
+ 0.0786 Batch*Epochs_4 15 + 0.0786 Batch*Epochs_8 5
- 0.0786 Batch*Epochs_8 15 - 0.2144 ROI*Epochs_64 5
+ 0.2144 ROI*Epochs_64 15 + 0.2144 ROI*Epochs_512 5
- 0.2144 ROI*Epochs_512 15 - 0.1335 LR*LRD*Batch_0.001 0.01 4
+ 0.1335 LR*LRD*Batch_0.001 0.01 8 + 0.1335 LR*LRD*Batch_0.001
0.10 4 - 0.1335 LR*LRD*Batch_0.001 0.10 8
+ 0.1335 LR*LRD*Batch_0.025 0.01 4 - 0.1335 LR*LRD*Batch_0.025
0.01 8 - 0.1335 LR*LRD*Batch_0.025 0.10 4
+ 0.1335 LR*LRD*Batch_0.025 0.10 8
- 0.2254 LR*LRDdecline*Batch_0.001 0.7 4
+ 0.2254 LR*LRDdecline*Batch_0.001 0.7 8
+ 0.2254 LR*LRDdecline*Batch_0.001 0.9 4
- 0.2254 LR*LRDdecline*Batch_0.001 0.9 8
+ 0.2254 LR*LRDdecline*Batch_0.025 0.7 4
- 0.2254 LR*LRDdecline*Batch_0.025 0.7 8
- 0.2254 LR*LRDdecline*Batch_0.025 0.9 4
+ 0.2254 LR*LRDdecline*Batch_0.025 0.9 8
+ 0.4060 LR*Batch*ROI_0.001 4 64 - 0.4060 LR*Batch*ROI_0.001 4
512 - 0.4060 LR*Batch*ROI_0.001 8 64
+ 0.4060 LR*Batch*ROI_0.001 8 512 - 0.4060 LR*Batch*ROI_0.025
4 64 + 0.4060 LR*Batch*ROI_0.025 4 512
+ 0.4060 LR*Batch*ROI_0.025 8 64 - 0.4060 LR*Batch*ROI_0.025 8
512 + 0.0959 LRD*Batch*ROI_0.01 4 64
- 0.0959 LRD*Batch*ROI_0.01 4 512 - 0.0959 LRD*Batch*ROI_0.01
8 64 + 0.0959 LRD*Batch*ROI_0.01 8 512
- 0.0959 LRD*Batch*ROI_0.10 4 64 + 0.0959 LRD*Batch*ROI_0.10 4
512 + 0.0959 LRD*Batch*ROI_0.10 8 64
- 0.0959 LRD*Batch*ROI_0.10 8 512
+ 3.9393 LRD*Batch*Epochs_0.01 4 5
- 3.9393 LRD*Batch*Epochs_0.01 4 15
- 3.9393 LRD*Batch*Epochs_0.01 8 5
+ 3.9393 LRD*Batch*Epochs_0.01 8 15
- 3.9393 LRD*Batch*Epochs_0.10 4 5
+ 3.9393 LRD*Batch*Epochs_0.10 4 15
+ 3.9393 LRD*Batch*Epochs_0.10 8 5
- 3.9393 LRD*Batch*Epochs_0.10 8 15

```

Fits and Diagnostics for Unusual Observations

test		-evaluation				
Obs	for segm ap	Fit	Resid	Std Resid		
1	76.720	77.201	-0.481	-2.05	R	
11	80.201	80.723	-0.522	-2.23	R	
14	80.842	81.314	-0.472	-2.02	R	
18	81.410	80.723	0.686	2.93	R	
115	89.813	90.312	-0.499	-2.13	R	
125	69.411	70.198	-0.787	-3.36	R	
131	74.673	74.673	-0.000	*		X

R Large residual
X Unusual X

Dataset Factors General linear model of the Bounding Box vs Images per class, number of classes, type of dataset and interactions

DATA VARIABLES

General Linear Model: test -evaluation ap for bbox versus Images per class, Num Classes, Type Dataset

The following terms cannot be estimated and were removed:
Images per class*Num Classes*Type Dataset

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
Images per class	Fixed	2	1000, 1500
Num Classes	Fixed	2	5, 10
Type Dataset	Fixed	2	Lego, Machine parts

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Images per class	1	42.95	42.95	4.85	0.030
Num Classes	1	1134.98	1134.98	128.08	0.000
Type Dataset	1	50.86	50.86	5.74	0.018
Images per class*Num Classes	1	48.42	48.42	5.46	0.021
Images per class*Type Dataset	1	5.51	5.51	0.62	0.432
Num Classes*Type Dataset	1	1150.79	1150.79	129.86	0.000
Error	124	1098.84	8.86		
Total	130	4285.18			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
2.97685	74.36%	73.12%	71.21%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	82.458	0.365	225.74	0.000	
Images per class 1000	-0.733	0.333	-2.20	0.030	1.63
Num Classes 5	4.134	0.365	11.32	0.000	1.88
Type Dataset Lego	0.875	0.365	2.40	0.018	1.88
Images per class*Num Classes 1000 5	0.778	0.333	2.34	0.021	1.63
Images per class*Type Dataset 1000 Lego	0.262	0.333	0.79	0.432	1.63
Num Classes*Type Dataset 5 Lego	-4.163	0.365	-11.40	0.000	1.88

Regression Equation

test -evaluation ap for bbox = 82.458 - 0.733 Images per class_1000
+ 0.733 Images per class_1500 + 4.134 Num Classes_5
- 4.134 Num Classes_10 + 0.875 Type Dataset_Lego
- 0.875 Type Dataset_Machine parts
+ 0.778 Images per class*Num Classes_1000 5
- 0.778 Images per class*Num Classes_1000 10
- 0.778 Images per class*Num Classes_1500 5
+ 0.778 Images per class*Num Classes_1500 10
+ 0.262 Images per class*Type Dataset_1000 Lego
- 0.262 Images per class*Type Dataset_1000 Machine parts
- 0.262 Images per class*Type Dataset_1500 Lego
+ 0.262 Images per class*Type Dataset_1500 Machine parts
- 4.163 Num Classes*Type Dataset_5 Lego
+ 4.163 Num Classes*Type Dataset_5 Machine parts
+ 4.163 Num Classes*Type Dataset_10 Lego
- 4.163 Num Classes*Type Dataset_10 Machine parts

Fits and Diagnostics for Unusual Observations

test					
-evaluation					
Obs	ap for bbox	Fit	Resid	Std Resid	
1	73.459	82.996	-9.537	-3.29	R
2	73.776	82.996	-9.220	-3.18	R
3	73.877	82.996	-9.119	-3.14	R
4	73.953	82.996	-9.044	-3.12	R
5	74.538	82.996	-8.459	-2.92	R

R Large residual

Dataset Factors General linear model of the Segmentation vs Images per class, number of classes, type of dataset and interactions

DATA VARIABLES

General Linear Model: test -evaluation for segm ap versus Images per class, Num Classes, Type Dataset

The following terms cannot be estimated and were removed:
Images per class*Num Classes*Type Dataset

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
Images per class	Fixed	2	1000, 1500
Num Classes	Fixed	2	5, 10
Type Dataset	Fixed	2	Lego, Machine parts

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Images per class	1	17.23	17.23	6.57	0.012
Num Classes	1	1075.22	1075.22	410.00	0.000
Type Dataset	1	0.47	0.47	0.18	0.672
Images per class*Num Classes	1	8.35	8.35	3.19	0.077
Images per class*Type Dataset	1	0.54	0.54	0.21	0.651
Num Classes*Type Dataset	1	1344.28	1344.28	512.59	0.000
Error	124	325.19	2.62		
Total	130	3762.00			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
1.61942	91.36%	90.94%	90.31%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	81.808	0.199	411.69	0.000	
Images per class_1000	-0.464	0.181	-2.56	0.012	1.63
Num Classes_5	4.024	0.199	20.25	0.000	1.88
Type Dataset_Lego	0.084	0.199	0.42	0.672	1.88
Images per class*Num Classes_1000_5	0.323	0.181	1.78	0.077	1.63
Images per class*Type Dataset_1000_Lego	0.082	0.181	0.45	0.651	1.63
Num Classes*Type Dataset_5_Lego	-4.499	0.199	-22.64	0.000	1.88

Regression Equation

test -evaluation for segm ap = 81.808 - 0.464 Images per class_1000
+ 0.464 Images per class_1500 + 4.024 Num Classes_5
- 4.024 Num Classes_10 + 0.084 Type Dataset_Lego
- 0.084 Type Dataset_Machine parts
+ 0.323 Images per class*Num Classes_1000_5
- 0.323 Images per class*Num Classes_1000_10
- 0.323 Images per class*Num Classes_1500_5
+ 0.323 Images per class*Num Classes_1500_10
+ 0.082 Images per class*Type Dataset_1000_Lego
- 0.082 Images per class*Type Dataset_1000_Machine parts
- 0.082 Images per class*Type Dataset_1500_Lego
+ 0.082 Images per class*Type Dataset_1500_Machine parts
- 4.499 Num Classes*Type Dataset_5_Lego
+ 4.499 Num Classes*Type Dataset_5_Machine parts
+ 4.499 Num Classes*Type Dataset_10_Lego
- 4.499 Num Classes*Type Dataset_10_Machine parts

Fits and Diagnostics for Unusual Observations

test					
-evaluation					
Obs	for segm ap	Fit	Resid	Std Resid	
1	76.720	81.475	-4.755	-3.01	R
2	76.901	81.475	-4.574	-2.90	R
3	77.293	81.475	-4.183	-2.65	R
4	77.452	81.475	-4.023	-2.55	R
5	77.639	81.475	-3.836	-2.43	R
40	84.869	81.475	3.394	2.15	R

R Large residual

5 Class General linear models for bounding box vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions

5 CLASSES

General Linear Model: test -evaluation ap for bbox versus LR, LRD, LRDecline, Batch, ROI, Images per class, Type Dataset, Epoc

The following terms cannot be estimated and were removed:
LR*LRD, LR*Images per class, LRD*LRDecline, LRD*Batch, LRD*Images per class, LRD*Type Dataset, LRD*Epochs, LRDecline*Batch, LRDecline*ROI, LRDecline*Images per class, LRDecline*Type Dataset, Batch*ROI, Batch*Images per class, Batch*Type Dataset, Batch*Epochs, ROI*Images per class, ROI*Type Dataset, ROI*Epochs, Images per class*Type Dataset, Images per class*Epochs, Type Dataset*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Type Dataset	Fixed	2	Lego, Machine parts
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	57.75	57.748	796.44	0.000
LRD	1	29.75	29.748	410.28	0.000
LRDecline	1	14.08	14.077	194.14	0.000
Batch	1	21.23	21.227	292.76	0.000
ROI	1	208.33	208.331	2873.22	0.000
Images per class	1	0.16	0.164	2.26	0.138
Type Dataset	1	864.61	864.612	11924.35	0.000
Epochs	1	123.82	123.821	1707.69	0.000
LR*LRDecline	1	5.51	5.509	75.97	0.000
LR*Batch	1	43.38	43.385	598.34	0.000
LR*ROI	1	131.65	131.651	1815.67	0.000
LR*Type Dataset	1	12.57	12.565	173.30	0.000
LR*Epochs	1	42.18	42.182	581.75	0.000
LRD*ROI	1	14.06	14.059	193.89	0.000
LRDecline*Epochs	1	5.54	5.536	76.35	0.000
Error	64	4.64	0.073		
Total	79	1579.26			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.269273	99.71%	99.64%	99.54%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	86.5915	0.0301	2876.25	0.000	
LR					
0.001	0.8496	0.0301	28.22	0.000	1.00
LRD					
0.01	-0.6098	0.0301	-20.26	0.000	1.00
LRDecline					
0.7	-0.4195	0.0301	-13.93	0.000	1.00
Batch					
4	-0.5151	0.0301	-17.11	0.000	1.00
ROI					
64	-1.6137	0.0301	-53.60	0.000	1.00
Images per class					
1000	0.0452	0.0301	1.50	0.138	1.00
Type Dataset					
Lego	-3.2875	0.0301	-109.20	0.000	1.00
Epochs					
5	-1.2441	0.0301	-41.32	0.000	1.00
LR*LRDecline					
0.001 0.7	0.2624	0.0301	8.72	0.000	1.00
LR*Batch					
0.001 4	0.7364	0.0301	24.46	0.000	1.00
LR*ROI					
0.001 64	1.2828	0.0301	42.61	0.000	1.00
LR*Type Dataset					
0.001 Lego	0.3963	0.0301	13.16	0.000	1.00
LR*Epochs					
0.001 5	0.7261	0.0301	24.12	0.000	1.00
LRD*ROI					

0.01 64	-0.4192	0.0301	-13.92	0.000	1.00
LRDdecline*Epochs					
0.7 5	-0.2631	0.0301	-8.74	0.000	1.00

Regression Equation

```
test -evaluation ap for bbox = 86.5915 + 0.8496 LR_0.001 - 0.8496 LR_0.025 - 0.6098 LRD_0.01
+ 0.6098 LRD_0.10 - 0.4195 LRDdecline_0.7
+ 0.4195 LRDdecline_0.9 - 0.5151 Batch_4 + 0.5151 Batch_8
- 1.6137 ROI_64 + 1.6137 ROI_512
+ 0.0452 Images per class_1000 - 0.0452 Images per class_1500
- 3.2875 Type Dataset_Lego + 3.2875 Type Dataset_Machine parts
- 1.2441 Epochs_5 + 1.2441 Epochs_15
+ 0.2624 LR*LRDdecline_0.001 0.7 - 0.2624 LR*LRDdecline_0.001
0.9 - 0.2624 LR*LRDdecline_0.025 0.7
+ 0.2624 LR*LRDdecline_0.025 0.9 + 0.7364 LR*Batch_0.001 4
- 0.7364 LR*Batch_0.001 8 - 0.7364 LR*Batch_0.025 4
+ 0.7364 LR*Batch_0.025 8 + 1.2828 LR*ROI_0.001 64
- 1.2828 LR*ROI_0.001 512 - 1.2828 LR*ROI_0.025 64
+ 1.2828 LR*ROI_0.025 512 - 0.3963 LR*Type Dataset_0.001 Lego
- 0.3963 LR*Type Dataset_0.001 Machine parts
- 0.3963 LR*Type Dataset_0.025 Lego
+ 0.3963 LR*Type Dataset_0.025 Machine parts
+ 0.7261 LR*Epochs_0.001 5 - 0.7261 LR*Epochs_0.001 15
- 0.7261 LR*Epochs_0.025 5 + 0.7261 LR*Epochs_0.025 15
- 0.4192 LRD*ROI_0.01 64 + 0.4192 LRD*ROI_0.01 512
+ 0.4192 LRD*ROI_0.10 64 - 0.4192 LRD*ROI_0.10 512
- 0.2631 LRDdecline*Epochs_0.7 5
+ 0.2631 LRDdecline*Epochs_0.7 15
+ 0.2631 LRDdecline*Epochs_0.9 5
- 0.2631 LRDdecline*Epochs_0.9 15
```

Fits and Diagnostics for Unusual Observations

Obs	test -evaluation ap for bbox	Fit	Resid	Std Resid	
5	74.538	73.921	0.617	2.56	R
18	84.210	83.638	0.572	2.38	R
31	85.842	86.328	-0.486	-2.02	R
74	89.585	89.095	0.490	2.03	R

R Large residual

5 Class General linear models for segmentation vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions

5 CLASSES

General Linear Model: test -evaluation for segm ap versus LR, LRD, LRDdecline, Batch, ROI, Images per class, Type Dataset, Epochs

The following terms cannot be estimated and were removed:
LR*LRD, LR*Images per class, LRD*LRDdecline, LRD*Batch, LRD*Images per class, LRD*Type Dataset, LRD*Epochs, LRDdecline*Batch, LRDdecline*ROI, LRDdecline*Images per class, LRDdecline*Type Dataset, Batch*ROI, Batch*Images per class, Batch*Type Dataset, Batch*Epochs, ROI*Images per class, ROI*Type Dataset, ROI*Epochs, Images per class*Type Dataset, Images per class*Epochs, Type Dataset*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDdecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Type Dataset	Fixed	2	Lego, Machine parts
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	0.00	0.00	0.03	0.855
LRD	1	5.44	5.44	71.65	0.000
LRDdecline	1	3.28	3.28	43.22	0.000
Batch	1	2.67	2.67	35.25	0.000
ROI	1	80.91	80.91	1066.59	0.000
Images per class	1	1.59	1.59	20.95	0.000
Type Dataset	1	1559.13	1559.13	20553.14	0.000
Epochs	1	47.27	47.27	623.11	0.000
LR*LRDdecline	1	0.54	0.54	7.12	0.010
LR*Batch	1	13.07	13.07	172.34	0.000
LR*ROI	1	34.22	34.22	451.12	0.000
LR*Type Dataset	1	1.12	1.12	14.81	0.000
LR*Epochs	1	11.66	11.66	153.76	0.000
LRD*ROI	1	3.18	3.18	41.91	0.000
LRDdecline*Epochs	1	0.65	0.65	8.60	0.005
Error	64	4.85	0.08		
Total	79	1769.60			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.275424	99.73%	99.66%	99.57%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	85.8311	0.0308	2787.33	0.000	
LR					
0.001	0.0057	0.0308	0.18	0.855	1.00
LRD					
0.01	-0.2606	0.0308	-8.46	0.000	1.00
LRDdecline					
0.7	-0.2024	0.0308	-6.57	0.000	1.00
Batch					
4	-0.1828	0.0308	-5.94	0.000	1.00
ROI					
64	-1.0057	0.0308	-32.66	0.000	1.00
Images per class					
1000	-0.1409	0.0308	-4.58	0.000	1.00
Type Dataset					
Lego	-4.4146	0.0308	-143.36	0.000	1.00
Epochs					
5	-0.7687	0.0308	-24.96	0.000	1.00
LR*LRDdecline					
0.001 0.7	0.0821	0.0308	2.67	0.010	1.00
LR*Batch					
0.001 4	0.4043	0.0308	13.13	0.000	1.00
LR*ROI					
0.001 64	0.6540	0.0308	21.24	0.000	1.00
LR*Type Dataset					
0.001 Lego	0.1185	0.0308	3.85	0.000	1.00
LR*Epochs					
0.001 5	0.3818	0.0308	12.40	0.000	1.00
LRD*ROI					

0.01 64	-0.1994	0.0308	-6.47	0.000	1.00
LRDdecline*Epochs					
0.7 5	-0.0903	0.0308	-2.93	0.005	1.00

Regression Equation

test -evaluation for segm ap = 85.8311 + 0.0057 LR_0.001 - 0.0057 LR_0.025 - 0.2606 LRD_0.01 + 0.2606 LRD_0.10 - 0.2024 LRDdecline_0.7 + 0.2024 LRDdecline_0.9 - 0.1828 Batch_4 + 0.1828 Batch_8 - 1.0057 ROI_64 + 1.0057 ROI_512 - 0.1409 Images per class_1000 + 0.1409 Images per class_1500 - 4.4146 Type Dataset_Lego + 4.4146 Type Dataset_Machine parts - 0.7687 Epochs_5 + 0.7687 Epochs_15 + 0.0821 LR*LRDdecline_0.001 0.7 - 0.0821 LR*LRDdecline_0.001 0.9 - 0.0821 LR*LRDdecline_0.025 0.7 + 0.0821 LR*LRDdecline_0.025 0.9 + 0.4043 LR*Batch_0.001 4 - 0.4043 LR*Batch_0.001 8 - 0.4043 LR*Batch_0.025 4 + 0.4043 LR*Batch_0.025 8 + 0.6540 LR*ROI_0.001 64 - 0.6540 LR*ROI_0.001 512 - 0.6540 LR*ROI_0.025 64 + 0.6540 LR*ROI_0.025 512 + 0.1185 LR*Type Dataset_0.001 Lego - 0.1185 LR*Type Dataset_0.001 Machine parts - 0.1185 LR*Type Dataset_0.025 Lego + 0.1185 LR*Type Dataset_0.025 Machine parts + 0.3818 LR*Epochs_0.001 5 - 0.3818 LR*Epochs_0.001 15 - 0.3818 LR*Epochs_0.025 5 + 0.3818 LR*Epochs_0.025 15 - 0.1994 LRD*ROI_0.01 64 + 0.1994 LRD*ROI_0.01 512 + 0.1994 LRD*ROI_0.10 64 - 0.1994 LRD*ROI_0.10 512 - 0.0903 LRDdecline*Epochs_0.7 5 + 0.0903 LRDdecline*Epochs_0.7 15 + 0.0903 LRDdecline*Epochs_0.9 5 - 0.0903 LRDdecline*Epochs_0.9 15

Fits and Diagnostics for Unusual Observations

Obs	test -evaluation for segm ap	Fit	Resid	Std Resid	
11	80.201	80.723	-0.522	-2.12	R
18	81.410	80.723	0.686	2.79	R
75	89.813	90.312	-0.499	-2.03	R

R Large residual

10 Class General linear models for bounding box vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions

10 CLASSES

General Linear Model: pipes bbox AP versus LR_1, LRD_1, LRDdecline_1, Batch_1, ROI_1, Images per class_1, Type Dataset_1, Epochs_1

The following terms cannot be estimated and were removed:

LR_1*LRD_1, LR_1*Images per class_1, LRD_1*LRDdecline_1, LRD_1*Batch_1, LRD_1*Images per class_1, LRD_1*Type Dataset_1, LRD_1*Epochs_1, LRDdecline_1*Batch_1, LRDdecline_1*ROI_1, LRDdecline_1*Images per class_1, LRDdecline_1*Type Dataset_1, Batch_1*ROI_1, Batch_1*Images per class_1, Batch_1*Type Dataset_1, Batch_1*Epochs_1, ROI_1*Images per class_1, ROI_1*Type Dataset_1, ROI_1*Epochs_1, Images per class_1*Type Dataset_1, Images per class_1*Epochs_1, Type Dataset_1*Epochs_1

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR_1	Fixed	2	0.001, 0.025
LRD_1	Fixed	2	0.01, 0.10
LRDdecline_1	Fixed	2	0.7, 0.9
Batch_1	Fixed	2	4, 8
ROI_1	Fixed	2	64, 512
Images per class_1	Fixed	2	1000, 1500
Type Dataset_1	Fixed	2	Lego, Machine parts
Epochs_1	Fixed	2	15, 5

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR_1	1	102.51	102.507	819.85	0.000
LRD_1	1	79.97	79.969	639.59	0.000
LRDdecline_1	1	6.48	6.485	51.86	0.000
Batch_1	1	6.36	6.364	50.90	0.000
ROI_1	1	245.93	245.930	1966.95	0.000
Images per class_1	1	71.94	71.943	575.40	0.000
Type Dataset_1	1	271.08	271.083	2168.12	0.000
Epochs_1	1	257.63	257.633	2060.55	0.000
LR_1*LRDdecline_1	1	7.20	7.198	57.57	0.000
LR_1*Batch_1	1	95.51	95.514	763.92	0.000
LR_1*ROI_1	1	356.54	356.543	2851.63	0.000
LR_1*Type Dataset_1	1	26.50	26.501	211.96	0.000
LR_1*Epochs_1	1	14.28	14.281	114.22	0.000
LRD_1*ROI_1	1	1.02	1.017	8.14	0.006
LRDdecline_1*Epochs_1	1	11.30	11.299	90.37	0.000
Error	64	8.00	0.125		
Total	79	1562.27			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.353598	99.49%	99.37%	99.20%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	81.5206	0.0395	2062.07	0.000	
LR_1					
0.001	1.1320	0.0395	28.63	0.000	1.00
LRD_1					
0.01	-0.9998	0.0395	-25.29	0.000	1.00
LRDdecline_1					
0.7	0.2847	0.0395	7.20	0.000	1.00
Batch_1					
4	-0.2820	0.0395	-7.13	0.000	1.00
ROI_1					
64	-1.7533	0.0395	-44.35	0.000	1.00
Images per class_1					
1000	-0.9483	0.0395	-23.99	0.000	1.00
Type Dataset_1					
Lego	1.8408	0.0395	46.56	0.000	1.00
Epochs_1					
15	1.7946	0.0395	45.39	0.000	1.00
LR_1*LRDdecline_1					
0.001 0.7	-0.3000	0.0395	-7.59	0.000	1.00
LR_1*Batch_1					
0.001 4	1.0927	0.0395	27.64	0.000	1.00
LR_1*ROI_1					
0.001 64	2.1111	0.0395	53.40	0.000	1.00
LR_1*Type Dataset_1					
0.001 Lego	-0.5756	0.0395	-14.56	0.000	1.00
LR_1*Epochs_1					
0.001 15	-0.4225	0.0395	-10.69	0.000	1.00
LRD_1*ROI_1					

0.01 64	-0.1128	0.0395	-2.85	0.006	1.00
LRDdecline_1*Epochs_1					
0.7 15	0.3758	0.0395	9.51	0.000	1.00

Regression Equation

pipes bbox AP = 81.5206 + 1.1320 LR_1_0.001 - 1.1320 LR_1_0.025 - 0.9998 LRD_1_0.01
 + 0.9998 LRD_1_0.10 + 0.2847 LRDdecline_1_0.7 - 0.2847 LRDdecline_1_0.9
 - 0.2820 Batch_1_4 + 0.2820 Batch_1_8 - 1.7533 ROI_1_64 + 1.7533 ROI_1_512
 - 0.9483 Images per class_1_1000 + 0.9483 Images per class_1_1500
 + 1.8408 Type Dataset_1_Lego - 1.8408 Type Dataset_1_Machine parts
 + 1.7946 Epochs_1_15 - 1.7946 Epochs_1_5 - 0.3000 LR_1*LRDdecline_1_0.001 0.7
 + 0.3000 LR_1*LRDdecline_1_0.001 0.9 + 0.3000 LR_1*LRDdecline_1_0.025 0.7
 - 0.3000 LR_1*LRDdecline_1_0.025 0.9 + 1.0927 LR_1*Batch_1_0.001 4
 - 1.0927 LR_1*Batch_1_0.001 8 - 1.0927 LR_1*Batch_1_0.025 4
 + 1.0927 LR_1*Batch_1_0.025 8 + 2.1111 LR_1*ROI_1_0.001 64
 - 2.1111 LR_1*ROI_1_0.001 512 - 2.1111 LR_1*ROI_1_0.025 64
 + 2.1111 LR_1*ROI_1_0.025 512 - 0.5756 LR_1*Type Dataset_1_0.001 Lego
 + 0.5756 LR_1*Type Dataset_1_0.001 Machine parts
 + 0.5756 LR_1*Type Dataset_1_0.025 Lego - 0.5756 LR_1*Type Dataset_1_0.025
 Machine parts - 0.4225 LR_1*Epochs_1_0.001 15 + 0.4225 LR_1*Epochs_1_0.001 5
 + 0.4225 LR_1*Epochs_1_0.025 15 - 0.4225 LR_1*Epochs_1_0.025 5
 - 0.1128 LRD_1*ROI_1_0.01 64 + 0.1128 LRD_1*ROI_1_0.01 512
 + 0.1128 LRD_1*ROI_1_0.10 64 - 0.1128 LRD_1*ROI_1_0.10 512
 + 0.3758 LRDdecline_1*Epochs_1_0.7 15 - 0.3758 LRDdecline_1*Epochs_1_0.7 5
 - 0.3758 LRDdecline_1*Epochs_1_0.9 15 + 0.3758 LRDdecline_1*Epochs_1_0.9 5

Fits and Diagnostics for Unusual Observations

Obs	pipes bbox AP	Fit	Resid	Std Resid	
2	71.565	72.368	-0.804	-2.54	R
18	73.124	72.368	0.756	2.39	R
26	71.519	72.368	-0.849	-2.68	R
35	70.562	71.413	-0.852	-2.69	R
37	82.713	81.804	0.909	2.87	R

R Large residual

10 Class General linear models for segmentation vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions

10 CLASSES

General Linear Model: pipes segm AP versus LR_1, LRD_1, LRDdecline_1, Batch_1, ROI_1, Images per class_1, Type Dataset_1, Epoc

The following terms cannot be estimated and were removed:

LR_1*LRD_1, LR_1*Images per class_1, LRD_1*LRDdecline_1, LRD_1*Batch_1, LRD_1*Images per class_1, LRD_1*Type Dataset_1, LRD_1*Epochs_1, LRDdecline_1*Batch_1, LRDdecline_1*ROI_1, LRDdecline_1*Images per class_1, LRDdecline_1*Type Dataset_1, Batch_1*ROI_1, Batch_1*Images per class_1, Batch_1*Type Dataset_1, Batch_1*Epochs_1, ROI_1*Images per class_1, ROI_1*Type Dataset_1, ROI_1*Epochs_1, Images per class_1*Type Dataset_1, Images per class_1*Epochs_1, Type Dataset_1*Epochs_1

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR_1	Fixed	2	0.001, 0.025
LRD_1	Fixed	2	0.01, 0.10
LRDdecline_1	Fixed	2	0.7, 0.9
Batch_1	Fixed	2	4, 8
ROI_1	Fixed	2	64, 512
Images per class_1	Fixed	2	1000, 1500
Type Dataset_1	Fixed	2	Lego, Machine parts
Epochs_1	Fixed	2	15, 5

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR_1	1	0.633	0.633	6.29	0.015
LRD_1	1	38.094	38.094	379.06	0.000
LRDdecline_1	1	0.042	0.042	0.42	0.521
Batch_1	1	0.058	0.058	0.58	0.450
ROI_1	1	73.569	73.569	732.06	0.000
Images per class_1	1	28.441	28.441	283.01	0.000
Type Dataset_1	1	95.717	95.717	952.44	0.000
Epochs_1	1	170.945	170.945	1701.00	0.000
LR_1*LRDdecline_1	1	0.948	0.948	9.44	0.003
LR_1*Batch_1	1	41.947	41.947	417.39	0.000
LR_1*ROI_1	1	166.772	166.772	1659.48	0.000
LR_1*Type Dataset_1	1	12.389	12.389	123.28	0.000
LR_1*Epochs_1	1	1.835	1.835	18.26	0.000
LRD_1*ROI_1	1	0.001	0.001	0.01	0.929
LRDdecline_1*Epochs_1	1	2.295	2.295	22.83	0.000
Error	64	6.432	0.100		
Total	79	640.118			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.317012	99.00%	98.76%	98.43%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	81.2734	0.0354	2293.07	0.000	
LR_1					
0.001	0.0889	0.0354	2.51	0.015	1.00
LRD_1					
0.01	-0.6901	0.0354	-19.47	0.000	1.00
LRDdecline_1					
0.7	0.0229	0.0354	0.65	0.521	1.00
Batch_1					
4	-0.0270	0.0354	-0.76	0.450	1.00
ROI_1					
64	-0.9590	0.0354	-27.06	0.000	1.00
Images per class_1					
1000	-0.5963	0.0354	-16.82	0.000	1.00
Type Dataset_1					
Lego	1.0938	0.0354	30.86	0.000	1.00
Epochs_1					
15	1.4618	0.0354	41.24	0.000	1.00
LR_1*LRDdecline_1					
0.001 0.7	-0.1089	0.0354	-3.07	0.003	1.00
LR_1*Batch_1					
0.001 4	0.7241	0.0354	20.43	0.000	1.00
LR_1*ROI_1					
0.001 64	1.4438	0.0354	40.74	0.000	1.00
LR_1*Type Dataset_1					
0.001 Lego	-0.3935	0.0354	-11.10	0.000	1.00
LR_1*Epochs_1					
0.001 15	-0.1515	0.0354	-4.27	0.000	1.00
LRD_1*ROI_1					

0.01 64	0.0032	0.0354	0.09	0.929	1.00
LRDdecline_1*Epochs_1					
0.7 15	0.1694	0.0354	4.78	0.000	1.00

Regression Equation

```

pipes segm AP = 81.2734 + 0.0889 LR_1_0.001 - 0.0889 LR_1_0.025 - 0.6901 LRD_1_0.01
+ 0.6901 LRD_1_0.10 + 0.0229 LRDdecline_1_0.7 - 0.0229 LRDdecline_1_0.9
- 0.0270 Batch_1_4 + 0.0270 Batch_1_8 - 0.9590 ROI_1_64 + 0.9590 ROI_1_512
- 0.5963 Images per class_1_1000 + 0.5963 Images per class_1_1500
+ 1.0938 Type Dataset_1_Lego - 1.0938 Type Dataset_1_Machine parts
+ 1.4618 Epochs_1_15 - 1.4618 Epochs_1_5 - 0.1089 LR_1*LRDdecline_1_0.001 0.7
+ 0.1089 LR_1*LRDdecline_1_0.001 0.9 + 0.1089 LR_1*LRDdecline_1_0.025 0.7
- 0.1089 LR_1*LRDdecline_1_0.025 0.9 + 0.7241 LR_1*Batch_1_0.001 4
- 0.7241 LR_1*Batch_1_0.001 8 - 0.7241 LR_1*Batch_1_0.025 4
+ 0.7241 LR_1*Batch_1_0.025 8 + 1.4438 LR_1*ROI_1_0.001 64
- 1.4438 LR_1*ROI_1_0.001 512 - 1.4438 LR_1*ROI_1_0.025 64
+ 1.4438 LR_1*ROI_1_0.025 512 - 0.3935 LR_1*Type Dataset_1_0.001 Lego
+ 0.3935 LR_1*Type Dataset_1_0.001 Machine parts
+ 0.3935 LR_1*Type Dataset_1_0.025 Lego - 0.3935 LR_1*Type Dataset_1_0.025
Machine parts - 0.1515 LR_1*Epochs_1_0.001 15 + 0.1515 LR_1*Epochs_1_0.001 5
+ 0.1515 LR_1*Epochs_1_0.025 15 - 0.1515 LR_1*Epochs_1_0.025 5
+ 0.0032 LRD_1*ROI_1_0.01 64 - 0.0032 LRD_1*ROI_1_0.01 512
- 0.0032 LRD_1*ROI_1_0.10 64 + 0.0032 LRD_1*ROI_1_0.10 512
+ 0.1694 LRDdecline_1*Epochs_1_0.7 15 - 0.1694 LRDdecline_1*Epochs_1_0.7 5
- 0.1694 LRDdecline_1*Epochs_1_0.9 15 + 0.1694 LRDdecline_1*Epochs_1_0.9 5

```

Fits and Diagnostics for Unusual Observations

Obs	pipes segm AP	Fit	Resid	Std Resid	
2	75.452	76.251	-0.798	-2.82	R
18	76.906	76.251	0.655	2.31	R
26	75.490	76.251	-0.761	-2.68	R
35	74.269	75.111	-0.842	-2.97	R
37	82.774	81.974	0.800	2.82	R

R Large residual

Pipe General linear models for Bounding Box vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Dataset, Epochs and Interactions

PIPE RESULTS

General Linear Model: test -evaluation ap for bbox versus LR, LRD, LRDecline, Batch, ROI, Images per class, Num Classes, Epochs

The following terms cannot be estimated and were removed:
LR*LRDecline, LR*Images per class, LRD*LRDecline, LRD*Batch, LRD*ROI, LRD*Images per class, LRD*Num Classes, LRDecline*Batch, LRDecline*Images per class, LRDecline*Num Classes, LRDecline*Epochs, Batch*ROI, Batch*Images per class, Batch*Num Classes, Batch*Epochs, ROI*Images per class, ROI*Num Classes, ROI*Epochs, Images per class*Num Classes, Images per class*Epochs, Num Classes*Epochs, LR*LRD*LRDecline, LR*LRD*Batch, LR*LRD*ROI, LR*LRD*Images per class, LR*LRD*Num Classes, LR*LRD*Epochs, LR*LRDecline*Batch, LR*LRDecline*ROI, LR*LRDecline*Images per class, LR*LRDecline*Num Classes, LR*LRDecline*Epochs, LR*Batch*ROI, LR*Batch*Images per class, LR*Batch*Num Classes, LR*Batch*Epochs, LR*ROI*Images per class, LR*ROI*Num Classes, LR*ROI*Epochs, LR*Images per class*Num Classes, LR*Images per class*Epochs, LR*Num Classes*Epochs, LRDecline*LRDecline*Batch, LRDecline*LRDecline*ROI, LRDecline*LRDecline*Images per class, LRDecline*LRDecline*Num Classes, LRDecline*LRDecline*Epochs, LRDecline*Batch*ROI, LRDecline*Batch*Images per class, LRDecline*Batch*Num Classes, LRDecline*Batch*Epochs, LRDecline*ROI*Images per class, LRDecline*ROI*Num Classes, LRDecline*ROI*Epochs, LRDecline*Images per class*Num Classes, LRDecline*Images per class*Epochs, LRDecline*Num Classes*Epochs, Batch*ROI*Num Classes, Batch*ROI*Epochs, Batch*Images per class*Num Classes, Batch*Images per class*Epochs, Batch*Num Classes*Epochs, ROI*Images per class*Num Classes, ROI*Images per class*Epochs, ROI*Num Classes*Epochs, Images per class*Num Classes*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Num Classes	Fixed	2	5, 10
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	93.38	93.38	795.59	0.000
LRD	1	16.40	16.40	139.69	0.000
LRDecline	1	5.51	5.51	46.92	0.000
Batch	1	0.50	0.50	4.27	0.043
ROI	1	187.69	187.69	1599.07	0.000
Images per class	1	14.98	14.98	127.65	0.000
Num Classes	1	2080.48	2080.48	17724.93	0.000
Epochs	1	299.06	299.06	2547.91	0.000
LR*LRD	1	3.72	3.72	31.68	0.000
LR*Batch	1	33.65	33.65	286.68	0.000
LR*ROI	1	311.16	311.16	2651.02	0.000
LR*Num Classes	1	31.46	31.46	268.04	0.000
LR*Epochs	1	33.19	33.19	282.78	0.000
LRD*Epochs	1	2.39	2.39	20.38	0.000
LRDecline*ROI	1	8.78	8.78	74.83	0.000
Error	64	7.51	0.12		
Total	79	3129.87			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.342601	99.76%	99.70%	99.62%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	84.7794	0.0383	2213.33	0.000	
LR					
0.001	1.0804	0.0383	28.21	0.000	1.00
LRD					
0.01	-0.4527	0.0383	-11.82	0.000	1.00
LRDecline					
0.7	-0.2624	0.0383	-6.85	0.000	1.00
Batch					
4	-0.0791	0.0383	-2.07	0.043	1.00
ROI					
64	-1.5317	0.0383	-39.99	0.000	1.00
Images per class					
1000	-0.4328	0.0383	-11.30	0.000	1.00
Num Classes					
5	5.0996	0.0383	133.14	0.000	1.00
Epochs					
5	-1.9335	0.0383	-50.48	0.000	1.00

LR*LRD					
0.001 0.01	0.2156	0.0383	5.63	0.000	1.00
LR*Batch					
0.001 4	0.6486	0.0383	16.93	0.000	1.00
LR*ROI					
0.001 64	1.9722	0.0383	51.49	0.000	1.00
LR*Num Classes					
0.001 5	-0.6271	0.0383	-16.37	0.000	1.00
LR*Epochs					
0.001 5	0.6441	0.0383	16.82	0.000	1.00
LRD*Epochs					
0.01 5	-0.1729	0.0383	-4.51	0.000	1.00
LRDdecline*ROI					
0.7 64	0.3313	0.0383	8.65	0.000	1.00

Regression Equation

test -evaluation ap for bbox = 84.7794 + 1.0804 LR_0.001 - 1.0804 LR_0.025 - 0.4527 LRD_0.01 + 0.4527 LRD_0.10 - 0.2624 LRDdecline_0.7 + 0.2624 LRDdecline_0.9 - 0.0791 Batch_4 + 0.0791 Batch_8 - 1.5317 ROI_64 + 1.5317 ROI_512 - 0.4328 Images per class_1000 + 0.4328 Images per class_1500 + 5.0996 Num Classes_5 - 5.0996 Num Classes_10 - 1.9335 Epochs_5 + 1.9335 Epochs_15 + 0.2156 LR*LRD_0.001 0.01 - 0.2156 LR*LRD_0.001 0.10 - 0.2156 LR*LRD_0.025 0.01 + 0.2156 LR*LRD_0.025 0.10 + 0.6486 LR*Batch_0.001 4 - 0.6486 LR*Batch_0.001 8 - 0.6486 LR*Batch_0.025 4 + 0.6486 LR*Batch_0.025 8 + 1.9722 LR*ROI_0.001 64 - 1.9722 LR*ROI_0.001 512 - 1.9722 LR*ROI_0.025 64 + 1.9722 LR*ROI_0.025 512 - 0.6271 LR*Num Classes_0.001 5 + 0.6271 LR*Num Classes_0.001 10 + 0.6271 LR*Num Classes_0.025 5 - 0.6271 LR*Num Classes_0.025 10 + 0.6441 LR*Epochs_0.001 5 - 0.6441 LR*Epochs_0.001 15 - 0.6441 LR*Epochs_0.025 5 + 0.6441 LR*Epochs_0.025 15 - 0.1729 LRD*Epochs_0.01 5 + 0.1729 LRD*Epochs_0.01 15 + 0.1729 LRD*Epochs_0.10 5 - 0.1729 LRD*Epochs_0.10 15 + 0.3313 LRDdecline*ROI_0.7 64 - 0.3313 LRDdecline*ROI_0.9 64 + 0.3313 LRDdecline*ROI_0.9 512

Fits and Diagnostics for Unusual Observations

Obs	test -evaluation ap for bbox	Fit	Resid	Std Resid
42	71.565	72.368	-0.804	-2.62 R
58	73.124	72.368	0.756	2.47 R
66	71.519	72.368	-0.849	-2.77 R
75	70.562	71.413	-0.852	-2.78 R
77	82.713	81.804	0.909	2.97 R

R Large residual

Pipe General linear models for Segmentation vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions

PIPE RESULTS

General Linear Model: test -evaluation for segm ap versus LR, LRD, LRDecline, Batch, ROI, Images per class, Num Classes, Epoc

The following terms cannot be estimated and were removed:

LR*LRDecline, LR*Images per class, LRD*LRDecline, LRD*Batch, LRD*ROI, LRD*Images per class, LRD*Num Classes, LRDecline*Batch, LRDecline*Images per class, LRDecline*Num Classes, LRDecline*Epochs, Batch*ROI, Batch*Images per class, Batch*Num Classes, Batch*Epochs, ROI*Images per class, ROI*Num Classes, ROI*Epochs, Images per class*Num Classes, Images per class*Epochs, Num Classes*Epochs, LR*LRD*LRDecline, LR*LRD*Batch, LR*LRD*ROI, LR*LRD*Images per class, LR*LRD*Num Classes, LR*LRD*Epochs, LR*LRDecline*Batch, LR*LRDecline*ROI, LR*LRDecline*Images per class, LR*LRDecline*Num Classes, LR*LRDecline*Epochs, LR*Batch*ROI, LR*Batch*Images per class, LR*Batch*Num Classes, LR*Batch*Epochs, LR*ROI*Images per class, LR*ROI*Num Classes, LR*ROI*Epochs, LR*Images per class*Num Classes, LR*Images per class*Epochs, LR*Num Classes*Epochs, LRD*LRDecline*Batch, LRD*LRDecline*ROI, LRD*LRDecline*Images per class, LRD*LRDecline*Num Classes, LRD*LRDecline*Epochs, LRD*Batch*ROI, LRD*Batch*Images per class, LRD*Batch*Num Classes, LRD*Batch*Epochs, LRD*ROI*Images per class, LRD*ROI*Num Classes, LRD*ROI*Epochs, LRD*Images per class*Num Classes, LRD*Images per class*Epochs, LRD*Num Classes*Epochs, LRDecline*Batch*ROI, LRDecline*Batch*Images per class, LRDecline*Batch*Num Classes, LRDecline*Batch*Epochs, LRDecline*ROI*Images per class, LRDecline*ROI*Num Classes, LRDecline*ROI*Epochs, LRDecline*Images per class*Num Classes, LRDecline*Images per class*Epochs, LRDecline*Num Classes*Epochs, Batch*ROI*Num Classes, Batch*ROI*Epochs, Batch*Images per class*Num Classes, Batch*Images per class*Epochs, Batch*Num Classes*Epochs, ROI*Images per class*Num Classes, ROI*Images per class*Epochs, ROI*Num Classes*Epochs, Images per class*Num Classes*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Num Classes	Fixed	2	5, 10
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	2.73	2.73	24.86	0.000
LRD	1	10.52	10.52	95.74	0.000
LRDecline	1	7.42	7.42	67.48	0.000
Batch	1	0.05	0.05	0.45	0.504
ROI	1	60.15	60.15	547.29	0.000
Images per class	1	10.09	10.09	91.84	0.000
Num Classes	1	2026.59	2026.59	18438.28	0.000
Epochs	1	182.44	182.44	1659.86	0.000
LR*LRD	1	1.40	1.40	12.71	0.001
LR*Batch	1	17.38	17.38	158.11	0.000
LR*ROI	1	155.79	155.79	1417.43	0.000
LR*Num Classes	1	7.09	7.09	64.49	0.000
LR*Epochs	1	4.74	4.74	43.09	0.000
LRD*Epochs	1	1.10	1.10	10.04	0.002
LRDecline*ROI	1	5.46	5.46	49.65	0.000
Error	64	7.03	0.11		
Total	79	2499.98			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.331530	99.72%	99.65%	99.56%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	85.2127	0.0371	2298.94	0.000	
LR					
0.001	0.1848	0.0371	4.99	0.000	1.00
LRD					
0.01	-0.3627	0.0371	-9.78	0.000	1.00
LRDecline					
0.7	-0.3045	0.0371	-8.21	0.000	1.00
Batch					
4	0.0249	0.0371	0.67	0.504	1.00
ROI					
64	-0.8671	0.0371	-23.39	0.000	1.00
Images per class					
1000	-0.3552	0.0371	-9.58	0.000	1.00
Num Classes					
5	5.0331	0.0371	135.79	0.000	1.00
Epochs					
5	-1.5101	0.0371	-40.74	0.000	1.00

LR*LRD					
0.001 0.01	0.1321	0.0371	3.57	0.001	1.00
LR*Batch					
0.001 4	0.4661	0.0371	12.57	0.000	1.00
LR*ROI					
0.001 64	1.3955	0.0371	37.65	0.000	1.00
LR*Num Classes					
0.001 5	-0.2977	0.0371	-8.03	0.000	1.00
LR*Epochs					
0.001 5	0.2433	0.0371	6.56	0.000	1.00
LRD*Epochs					
0.01 5	-0.1175	0.0371	-3.17	0.002	1.00
LRDdecline*ROI					
0.7 64	0.2612	0.0371	7.05	0.000	1.00

Regression Equation

test -evaluation for segm ap = 85.2127 + 0.1848 LR_0.001 - 0.1848 LR_0.025 - 0.3627 LRD_0.01 + 0.3627 LRD_0.10 - 0.3045 LRDdecline_0.7 + 0.3045 LRDdecline_0.9 + 0.0249 Batch_4 - 0.0249 Batch_8 - 0.8671 ROI_64 + 0.8671 ROI_512 - 0.3552 Images per class_1000 + 0.3552 Images per class_1500 + 5.0331 Num Classes_5 - 5.0331 Num Classes_10 - 1.5101 Epochs_5 + 1.5101 Epochs_15 + 0.1321 LR*LRD_0.001 0.01 - 0.1321 LR*LRD_0.001 0.10 - 0.1321 LR*LRD_0.025 0.01 + 0.1321 LR*LRD_0.025 0.10 + 0.4661 LR*Batch_0.001 4 - 0.4661 LR*Batch_0.001 8 - 0.4661 LR*Batch_0.025 4 + 0.4661 LR*Batch_0.025 8 + 1.3955 LR*ROI_0.001 64 - 1.3955 LR*ROI_0.001 512 - 1.3955 LR*ROI_0.025 64 + 1.3955 LR*ROI_0.025 512 - 0.2977 LR*Num Classes_0.001 5 + 0.2977 LR*Num Classes_0.001 10 + 0.2977 LR*Num Classes_0.025 5 - 0.2977 LR*Num Classes_0.025 10 + 0.2433 LR*Epochs_0.001 5 - 0.2433 LR*Epochs_0.001 15 - 0.2433 LR*Epochs_0.025 5 + 0.2433 LR*Epochs_0.025 15 - 0.1175 LRD*Epochs_0.01 5 + 0.1175 LRD*Epochs_0.01 15 + 0.1175 LRD*Epochs_0.10 5 - 0.1175 LRD*Epochs_0.10 15 + 0.2612 LRDdecline*ROI_0.7 64 - 0.2612 LRDdecline*ROI_0.9 64 + 0.2612 LRDdecline*ROI_0.9 512

Fits and Diagnostics for Unusual Observations

		test			
Obs	-evaluation	Fit	Resid	Std Resid	
42	for segm ap	75.452	76.251	-0.798	-2.69 R
58		76.906	76.251	0.655	2.21 R
66		75.490	76.251	-0.761	-2.57 R
75		74.269	75.111	-0.842	-2.84 R
77		82.774	81.974	0.800	2.70 R

R Large residual

Legos General linear models for Bounding Box vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Dataset, Epochs and Interactions

LEGO RESULTS

General Linear Model: test -evaluation ap for bbox versus LR, LRD, LRDecline, Batch, ROI, Images per class, Num Classes, Epochs

The following terms cannot be estimated and were removed:
LR*LRDecline, LR*Images per class, LRD*LRDecline, LRD*Batch, LRD*ROI, LRD*Images per class, LRD*Num Classes, LRDecline*Batch, LRDecline*Images per class, LRDecline*Num Classes, LRDecline*Epochs, Batch*ROI, Batch*Images per class, Batch*Num Classes, Batch*Epochs, ROI*Images per class, ROI*Num Classes, ROI*Epochs, Images per class*Num Classes, Images per class*Epochs, Num Classes*Epochs, LR*LRD*LRDecline, LR*LRD*Batch, LR*LRD*ROI, LR*LRD*Images per class, LR*LRD*Num Classes, LR*LRD*Epochs, LR*LRDecline*Batch, LR*LRDecline*ROI, LR*LRDecline*Images per class, LR*LRDecline*Num Classes, LR*LRDecline*Epochs, LR*Batch*ROI, LR*Batch*Images per class, LR*Batch*Num Classes, LR*Batch*Epochs, LR*ROI*Images per class, LR*ROI*Num Classes, LR*ROI*Epochs, LR*Images per class*Num Classes, LR*Images per class*Epochs, LR*Num Classes*Epochs, LRDecline*LRDecline*Batch, LRDecline*LRDecline*ROI, LRDecline*LRDecline*Images per class, LRDecline*LRDecline*Num Classes, LRDecline*LRDecline*Epochs, LRDecline*Batch*ROI, LRDecline*Batch*Images per class, LRDecline*Batch*Num Classes, LRDecline*Batch*Epochs, LRDecline*ROI*Images per class, LRDecline*ROI*Num Classes, LRDecline*ROI*Epochs, LRDecline*Images per class*Num Classes, LRDecline*Images per class*Epochs, LRDecline*Num Classes*Epochs, Batch*ROI*Images per class, Batch*ROI*Num Classes, Batch*ROI*Epochs, Batch*Images per class*Num Classes, Batch*Images per class*Epochs, Batch*Num Classes*Epochs, ROI*Images per class*Num Classes, ROI*Images per class*Epochs, ROI*Num Classes*Epochs, Images per class*Num Classes*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Num Classes	Fixed	2	5, 10
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	64.969	64.969	810.46	0.000
LRD	1	107.072	107.072	1335.67	0.000
LRDecline	1	1.303	1.303	16.25	0.000
Batch	1	41.243	41.243	514.49	0.000
ROI	1	269.477	269.477	3361.59	0.000
Images per class	1	17.696	17.696	220.75	0.000
Num Classes	1	0.066	0.066	0.82	0.368
Epochs	1	97.714	97.714	1218.93	0.000
LR*LRD	1	48.416	48.416	603.97	0.000
LR*Batch	1	111.493	111.493	1390.81	0.000
LR*ROI	1	161.707	161.707	2017.21	0.000
LR*Num Classes	1	9.509	9.509	118.62	0.000
LR*Epochs	1	20.364	20.364	254.03	0.000
LRD*Epochs	1	0.289	0.289	3.61	0.062
LRDecline*ROI	1	0.050	0.050	0.62	0.434
Error	64	5.130	0.080		
Total	79	956.499			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.283132	99.46%	99.34%	99.16%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	83.3327	0.0317	2632.52	0.000	
LR					
0.001	0.9012	0.0317	28.47	0.000	1.00
LRD					
0.01	-1.1569	0.0317	-36.55	0.000	1.00
LRDecline					
0.7	0.1276	0.0317	4.03	0.000	1.00
Batch					
4	-0.7180	0.0317	-22.68	0.000	1.00
ROI					
64	-1.8353	0.0317	-57.98	0.000	1.00
Images per class					
1000	-0.4703	0.0317	-14.86	0.000	1.00
Num Classes					
5	-0.0287	0.0317	-0.91	0.368	1.00
Epochs					
5	-1.1052	0.0317	-34.91	0.000	1.00

LR*LRD					
0.001 0.01	0.7779	0.0317	24.58	0.000	1.00
LR*Batch					
0.001 4	1.1805	0.0317	37.29	0.000	1.00
LR*ROI					
0.001 64	1.4217	0.0317	44.91	0.000	1.00
LR*Num Classes					
0.001 5	0.3448	0.0317	10.89	0.000	1.00
LR*Epochs					
0.001 5	0.5045	0.0317	15.94	0.000	1.00
LRD*Epochs					
0.01 5	-0.0602	0.0317	-1.90	0.062	1.00
LRDdecline*ROI					
0.7 64	0.0249	0.0317	0.79	0.434	1.00

Regression Equation

test -evaluation ap for bbox = 83.3327 + 0.9012 LR_0.001 - 0.9012 LR_0.025 - 1.1569 LRD_0.01 + 1.1569 LRD_0.10 + 0.1276 LRDdecline_0.7 - 0.1276 LRDdecline_0.9 - 0.7180 Batch_4 + 0.7180 Batch_8 - 1.8353 ROI_64 + 1.8353 ROI_512 - 0.4703 Images per class_1000 + 0.4703 Images per class_1500 - 0.0287 Num Classes_5 + 0.0287 Num Classes_10 - 1.1052 Epochs_5 + 1.1052 Epochs_15 + 0.7779 LR*LRD_0.001 0.01 - 0.7779 LR*LRD_0.001 0.10 - 0.7779 LR*LRD_0.025 0.01 + 0.7779 LR*LRD_0.025 0.10 + 1.1805 LR*Batch_0.001 4 - 1.1805 LR*Batch_0.001 8 - 1.1805 LR*Batch_0.025 4 + 1.1805 LR*Batch_0.025 8 + 1.4217 LR*ROI_0.001 64 - 1.4217 LR*ROI_0.001 512 - 1.4217 LR*ROI_0.025 64 + 1.4217 LR*ROI_0.025 512 + 0.3448 LR*Num Classes_0.001 5 - 0.3448 LR*Num Classes_0.001 10 - 0.3448 LR*Num Classes_0.025 5 + 0.3448 LR*Num Classes_0.025 10 + 0.5045 LR*Epochs_0.001 5 - 0.5045 LR*Epochs_0.025 5 + 0.5045 LR*Epochs_0.025 15 - 0.0602 LRD*Epochs_0.01 5 + 0.0602 LRD*Epochs_0.01 15 + 0.0602 LRD*Epochs_0.10 5 - 0.0602 LRD*Epochs_0.10 15 + 0.0249 LRDdecline*ROI_0.7 64 - 0.0249 LRDdecline*ROI_0.9 512 + 0.0249 LRDdecline*ROI_0.9 512

Fits and Diagnostics for Unusual Observations

	test				
Obs	-evaluation	Fit	Resid	Std	
ap for bbox				Resid	
5	74.538	73.921	0.617	2.44	R
18	84.210	83.638	0.572	2.26	R
59	77.281	76.762	0.520	2.05	R

R Large residual

Legos General linear models for Segmentation vs. LR, LRD, LRDecline, Batch, ROI, Images per class, Type of Datataset, Epochs and Interactions

LEGO RESULTS

General Linear Model: test -evaluation for segm ap versus LR, LRD, LRDecline, Batch, ROI, Images per class, Num Classes, Epoc

The following terms cannot be estimated and were removed:
LR*LRDecline, LR*Images per class, LRD*LRDecline, LRD*Batch, LRD*ROI, LRD*Images per class, LRD*Num Classes, LRDecline*Batch, LRDecline*Images per class, LRDecline*Num Classes, LRDecline*Epochs, Batch*ROI, Batch*Images per class, Batch*Num Classes, Batch*Epochs, ROI*Images per class, ROI*Num Classes, ROI*Epochs, Images per class*Num Classes, Images per class*Epochs, Num Classes*Epochs, LR*LRD*LRDecline, LR*LRD*Batch, LR*LRD*ROI, LR*LRD*Images per class, LR*LRD*Num Classes, LR*LRD*Epochs, LR*LRDecline*Batch, LR*LRDecline*ROI, LR*LRDecline*Images per class, LR*LRDecline*Num Classes, LR*LRDecline*Epochs, LR*Batch*ROI, LR*Batch*Images per class, LR*Batch*Num Classes, LR*Batch*Epochs, LR*ROI*Images per class, LR*ROI*Num Classes, LR*ROI*Epochs, LR*Images per class*Num Classes, LR*Images per class*Epochs, LR*Num Classes*Epochs, LRD*LRDecline*Batch, LRD*LRDecline*ROI, LRD*LRDecline*Images per class, LRD*LRDecline*Num Classes, LRD*LRDecline*Epochs, LRD*Batch*ROI, LRD*Batch*Images per class, LRD*Batch*Num Classes, LRD*Batch*Epochs, LRD*ROI*Images per class, LRD*ROI*Num Classes, LRD*ROI*Epochs, LRD*Images per class*Num Classes, LRD*Images per class*Epochs, LRD*Num Classes*Epochs, LRDecline*Batch*ROI, LRDecline*Batch*Images per class, LRDecline*Batch*Num Classes, LRDecline*Batch*Epochs, LRDecline*ROI*Images per class, LRDecline*ROI*Num Classes, LRDecline*ROI*Epochs, LRDecline*Images per class*Num Classes, LRDecline*Images per class*Epochs, LRDecline*Num Classes*Epochs, Batch*ROI*Num Classes, Batch*ROI*Epochs, Batch*Images per class*Num Classes, Batch*Images per class*Epochs, Batch*Num Classes*Epochs, ROI*Images per class*Num Classes, ROI*Images per class*Epochs, ROI*Num Classes*Epochs, Images per class*Num Classes*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 8
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Num Classes	Fixed	2	5, 10
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	0.651	0.6510	9.80	0.003
LRD	1	27.661	27.6612	416.31	0.000
LRDecline	1	1.248	1.2484	18.79	0.000
Batch	1	4.407	4.4073	66.33	0.000
ROI	1	96.361	96.3609	1450.28	0.000
Images per class	1	11.672	11.6719	175.67	0.000
Num Classes	1	18.077	18.0768	272.06	0.000
Epochs	1	41.510	41.5098	624.74	0.000
LR*LRD	1	8.355	8.3550	125.75	0.000
LR*Batch	1	35.089	35.0889	528.10	0.000
LR*ROI	1	39.466	39.4663	593.99	0.000
LR*Num Classes	1	3.677	3.6774	55.35	0.000
LR*Epochs	1	6.728	6.7284	101.27	0.000
LRD*Epochs	1	0.118	0.1180	1.78	0.187
LRDecline*ROI	1	0.275	0.2754	4.14	0.046
Error	64	4.252	0.0664		
Total	79	299.549			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.257766	98.58%	98.25%	97.78%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	81.8919	0.0288	2841.58	0.000	
LR					
0.001	-0.0902	0.0288	-3.13	0.003	1.00
LRD					
0.01	-0.5880	0.0288	-20.40	0.000	1.00
LRDecline					
0.7	0.1249	0.0288	4.33	0.000	1.00
Batch					
4	-0.2347	0.0288	-8.14	0.000	1.00
ROI					
64	-1.0975	0.0288	-38.08	0.000	1.00
Images per class					
1000	-0.3820	0.0288	-13.25	0.000	1.00
Num Classes					
5	-0.4754	0.0288	-16.49	0.000	1.00
Epochs					
5	-0.7203	0.0288	-24.99	0.000	1.00

LR*LRD					
0.001 0.01	0.3232	0.0288	11.21	0.000	1.00
LR*Batch					
0.001 4	0.6623	0.0288	22.98	0.000	1.00
LR*ROI					
0.001 64	0.7024	0.0288	24.37	0.000	1.00
LR*Num Classes					
0.001 5	0.2144	0.0288	7.44	0.000	1.00
LR*Epochs					
0.001 5	0.2900	0.0288	10.06	0.000	1.00
LRD*Epochs					
0.01 5	-0.0384	0.0288	-1.33	0.187	1.00
LRDdecline*ROI					
0.7 64	0.0587	0.0288	2.04	0.046	1.00

Regression Equation

test -evaluation for segm ap = 81.8919 - 0.0902 LR_0.001 + 0.0902 LR_0.025 - 0.5880 LRD_0.01 + 0.5880 LRD_0.10 + 0.1249 LRDdecline_0.7 - 0.1249 LRDdecline_0.9 - 0.2347 Batch_4 + 0.2347 Batch_8 - 1.0975 ROI_64 + 1.0975 ROI_512 - 0.3820 Images per class_1000 + 0.3820 Images per class_1500 - 0.4754 Num Classes_5 + 0.4754 Num Classes_10 - 0.7203 Epochs_5 + 0.7203 Epochs_15 + 0.3232 LR*LRD_0.001 0.01 - 0.3232 LR*LRD_0.001 0.10 - 0.3232 LR*LRD_0.025 0.01 + 0.3232 LR*LRD_0.025 0.10 + 0.6623 LR*Batch_0.001 4 - 0.6623 LR*Batch_0.001 8 - 0.6623 LR*Batch_0.025 4 + 0.6623 LR*Batch_0.025 8 + 0.7024 LR*ROI_0.001 64 - 0.7024 LR*ROI_0.001 512 - 0.7024 LR*ROI_0.025 64 + 0.7024 LR*ROI_0.025 512 + 0.2144 LR*Num Classes_0.001 5 - 0.2144 LR*Num Classes_0.001 10 - 0.2144 LR*Num Classes_0.025 5 + 0.2144 LR*Num Classes_0.025 10 + 0.2900 LR*Epochs_0.001 5 - 0.2900 LR*Epochs_0.001 15 - 0.2900 LR*Epochs_0.025 5 + 0.2900 LR*Epochs_0.025 15 - 0.0384 LRD*Epochs_0.01 5 + 0.0384 LRD*Epochs_0.01 15 + 0.0384 LRD*Epochs_0.10 5 - 0.0384 LRD*Epochs_0.10 15 + 0.0587 LRDdecline*ROI_0.7 64 - 0.0587 LRDdecline*ROI_0.9 512 + 0.0587 LRDdecline*ROI_0.9 512

Fits and Diagnostics for Unusual Observations

Obs	test -evaluation for segm ap	Fit	Resid	Std Resid
1	76.720	77.201	-0.481	-2.09 R
11	80.201	80.723	-0.522	-2.27 R
14	80.842	81.314	-0.472	-2.05 R
18	81.410	80.723	0.686	2.98 R

R Large residual

APPENDIX H

Two Sample T-Tests

5 Class Two-Sample T-Test Bounding Box Results

5 CLASSES

General Linear Model: test -evaluation ap for bbox versus LR, LRD, LRDdecline, Batch, ROI, Images per class, Type Dataset, Epoch

The following terms cannot be estimated and were removed:
LR*LRD, LR*Images per class, LRD*LRDdecline, LRD*Batch, LRD*Images per class, LRD*Type Dataset, LRD*Epochs, LRDdecline*Batch, LRDdecline*ROI, LRDdecline*Images per class, LRDdecline*Type Dataset, Batch*ROI, Batch*Images per class, Batch*Type Dataset, Batch*Epochs, ROI*Images per class, ROI*Type Dataset, ROI*Epochs, Images per class*Type Dataset, Images per class*Epochs, Type Dataset*Epochs

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
LR	Fixed	2	0.001, 0.025
LRD	Fixed	2	0.01, 0.10
LRDdecline	Fixed	2	0.7, 0.9
Batch	Fixed	2	4, 6
ROI	Fixed	2	64, 512
Images per class	Fixed	2	1000, 1500
Type Dataset	Fixed	2	Lego, Machine parts
Epochs	Fixed	2	5, 15

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
LR	1	57.75	57.748	796.44	0.000
LRD	1	29.75	29.748	410.28	0.000
LRDdecline	1	14.08	14.077	194.14	0.000
Batch	1	21.23	21.227	292.76	0.000
ROI	1	208.33	208.331	2873.22	0.000
Images per class	1	0.16	0.164	2.26	0.138
Type Dataset	1	864.61	864.612	11924.35	0.000
Epochs	1	123.82	123.821	1707.69	0.000
LR*LRDdecline	1	5.51	5.509	75.97	0.000
LR*Batch	1	43.38	43.385	598.34	0.000
LR*ROI	1	131.65	131.651	1815.67	0.000
LR*Type Dataset	1	12.57	12.565	173.30	0.000
LR*Epochs	1	42.18	42.182	581.75	0.000
LRD*ROI	1	14.06	14.059	193.89	0.000
LRDdecline*Epochs	1	5.54	5.536	76.35	0.000
Error	64	4.64	0.073		
Total	79	1579.26			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.269273	99.71%	99.64%	99.54%

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	86.5915	0.0301	2876.25	0.000	
LR					
0.001	0.8496	0.0301	28.22	0.000	1.00
LRD					
0.01	-0.6098	0.0301	-20.26	0.000	1.00
LRDdecline					
0.7	-0.4195	0.0301	-13.93	0.000	1.00
Batch					
4	-0.5151	0.0301	-17.11	0.000	1.00
ROI					
64	-1.6137	0.0301	-53.60	0.000	1.00
Images per class					
1000	0.0452	0.0301	1.50	0.138	1.00
Type Dataset					
Lego	-3.2875	0.0301	-109.20	0.000	1.00
Epochs					
5	-1.2441	0.0301	-41.32	0.000	1.00
LR*LRDdecline					
0.001 0.7	0.2624	0.0301	8.72	0.000	1.00
LR*Batch					
0.001 4	0.7364	0.0301	24.46	0.000	1.00
LR*ROI					
0.001 64	1.2828	0.0301	42.61	0.000	1.00
LR*Type Dataset					
0.001 Lego	0.3963	0.0301	13.16	0.000	1.00
LR*Epochs					
0.001 5	0.7261	0.0301	24.12	0.000	1.00
LRD*ROI					

0.01 64	-0.4192	0.0301	-13.92	0.000	1.00
LRDdecline*Epochs					
0.7 5	-0.2631	0.0301	-8.74	0.000	1.00

Regression Equation

test -evaluation ap for bbox = 86.5915 + 0.8496 LR_0.001 - 0.8496 LR_0.025 - 0.6098 LRD_0.01 + 0.6098 LRD_0.10 - 0.4195 LRDdecline_0.7 + 0.4195 LRDdecline_0.9 - 0.5151 Batch_4 + 0.5151 Batch_8 - 1.6137 ROI_64 + 1.6137 ROI_512 + 0.0452 Images per class_1000 - 0.0452 Images per class_1500 - 3.2875 Type Dataset_Lego + 3.2875 Type Dataset_Machine parts - 1.2441 Epochs_3 + 1.2441 Epochs_15 + 0.2624 LR*LRDdecline_0.001 0.7 - 0.2624 LR*LRDdecline_0.001 0.9 - 0.2624 LR*LRDdecline_0.025 0.7 + 0.2624 LR*LRDdecline_0.025 0.9 + 0.7364 LR*Batch_0.001 4 - 0.7364 LR*Batch_0.001 8 - 0.7364 LR*Batch_0.025 4 + 0.7364 LR*Batch_0.025 8 + 1.2828 LR*ROI_0.001 64 - 1.2828 LR*ROI_0.001 512 - 1.2828 LR*ROI_0.025 64 + 1.2828 LR*ROI_0.025 512 + 0.3963 LR*Type Dataset_0.001 Lego - 0.3963 LR*Type Dataset_0.001 Machine parts - 0.3963 LR*Type Dataset_0.025 Lego + 0.3963 LR*Type Dataset_0.025 Machine parts + 0.7261 LR*Epochs_0.001 5 - 0.7261 LR*Epochs_0.001 15 - 0.7261 LR*Epochs_0.025 5 + 0.7261 LR*Epochs_0.025 15 - 0.4192 LRD*ROI_0.01 64 + 0.4192 LRD*ROI_0.01 512 + 0.4192 LRD*ROI_0.10 64 - 0.4192 LRD*ROI_0.10 512 - 0.2631 LRDdecline*Epochs_0.7 5 + 0.2631 LRDdecline*Epochs_0.7 15 + 0.2631 LRDdecline*Epochs_0.9 5 - 0.2631 LRDdecline*Epochs_0.9 15

Fits and Diagnostics for Unusual Observations

Obs	test -evaluation ap for bbox	Fit	Resid	Std Resid	
5	74.538	73.921	0.617	2.56	R
18	84.210	83.638	0.572	2.38	R
31	85.842	86.328	-0.486	-2.02	R
74	89.585	89.095	0.490	2.03	R

R Large residual

10 Class Two-Sample T-Test Bounding Box Results

10 CLASSES

Two-Sample T-Test and CI: pipes bbox AP, lego bbox AP

Method

μ_1 : mean of pipes bbox AP
 μ_2 : mean of lego bbox AP
Difference: $\mu_1 - \mu_2$

Equal variances are not assumed for this analysis.

Descriptive Statistics

Sample	N	Mean	StDev	SE Mean
pipes bbox AP	40	79.68	5.06	0.80
lego bbox AP	40	83.36	2.74	0.43

Estimation for Difference

Difference	95% CI for Difference
-3.682	(-5.501, -1.862)

Test

Null hypothesis $H_0: \mu_1 - \mu_2 = 0$
Alternative hypothesis $H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
-4.05	60	0.000

5 Class Two-Sample T-Test Segmentation Results

5 CLASSES

Two-Sample T-Test and CI: lego segm ap, pipes segm AP

Method

μ_1 : mean of lego segm ap
 μ_2 : mean of pipes segm AP
Difference: $\mu_1 - \mu_2$

Equal variances are not assumed for this analysis.

Descriptive Statistics

Sample	N	Mean	StDev	SE Mean
lego segm ap	40	81.42	2.19	0.35
pipes segm AP	40	90.246	0.762	0.12

Estimation for Difference

Difference	95% CI for Difference
-8.829	(-9.568, -8.091)

Test

Null hypothesis $H_0: \mu_1 - \mu_2 = 0$
Alternative hypothesis $H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
-24.04	48	0.000

10 Class Two-Sample T-Test Segmentation Results

10 CLASSES

Two-Sample T-Test and CI: pipes segm AP, lego segm ap

Method

μ_1 : mean of pipes segm AP

μ_2 : mean of lego segm ap

Difference: $\mu_1 - \mu_2$

Equal variances are not assumed for this analysis.

Descriptive Statistics

Sample	N	Mean	StDev	SE Mean
pipes segm AP	40	80.18	3.40	0.54
lego segm ap	40	82.37	1.55	0.25

Estimation for Difference

Difference	95% CI for Difference
-2.188	(-3.372, -1.003)

Test

Null hypothesis

$H_0: \mu_1 - \mu_2 = 0$

Alternative hypothesis

$H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
-3.70	54	0.001

APPENDIX I

One-Way ANOVA and Descriptive Statics

One-Way ANOVA Bounding Box vs. Experiments Pipe Dataset

PIPE RESULTS

One-way ANOVA: test -evaluation ap for bbox versus actual iterations

* NOTE * Cannot draw the interval plot for the Games-Howell procedure. Interval plots for comparisons are illegible with more than 45 intervals.

Method

Null hypothesis All means are equal
Alternative hypothesis Not all means are equal
Significance level $\alpha = 0.05$
Rows unused 176

Equal variances were not assumed for the analysis.

Factor Information

Factor	Levels	Values
actual iterations	16	3117, 4677, 6244, 6245, 9365, 9366, 9369, 12497, 14053, 18741, 18745, 18746, 28117, 28121, 37497, 56245

Welch's Test

Source	DF Num	DF Den	F-Value	P-Value
actual iterations	15	23.1212	7602.75	0.000

Model Summary

R-sq	R-sq(adj)	R-sq(pred)
93.97%	92.55%	91.28%

Means

actual iterations	N	Mean	StDev	95% CI
3117	5	90.8944	0.1747	(90.6775, 91.1114)
4677	5	89.8258	0.0678	(89.7416, 89.9100)
6244	5	71.413	0.540	(70.743, 72.084)
6245	5	89.141	0.237	(88.846, 89.436)
9365	5	90.7085	0.1585	(90.5117, 90.9053)
9366	5	78.5171	0.0849	(78.4117, 78.6225)
9369	5	89.810	0.250	(89.499, 90.121)
12497	5	80.7979	0.0240	(80.7681, 80.8277)
14053	5	89.095	0.450	(88.537, 89.654)
18741	5	82.110	0.304	(81.733, 82.488)
18745	6	85.44	6.04	(79.11, 91.77)
18746	4	72.179	0.740	(71.002, 73.356)
28117	5	86.303	0.229	(86.020, 86.587)
28121	5	91.6538	0.1242	(91.4996, 91.8081)
37497	5	81.804	0.595	(81.066, 82.543)
56245	5	84.124	0.237	(83.830, 84.418)

Games-Howell Pairwise Comparisons

Grouping Information Using the Games-Howell Method and 95% Confidence

actual iterations	N	Mean	Grouping
28121	5	91.6538	A
3117	5	90.8944	B
9365	5	90.7085	B
4677	5	89.8258	C
9369	5	89.810	C D
6245	5	89.141	D
14053	5	89.095	C D
28117	5	86.303	E
18745	6	85.44	A B C D E F G H I J K
56245	5	84.124	F
18741	5	82.110	K
37497	5	81.804	J K
12497	5	80.7979	I J
9366	5	78.5171	H
18746	4	72.179	G L
6244	5	71.413	L

Means that do not share a letter are significantly different.

Descriptive Statics Bounding Box and Segmentation AP Pipe Dataset

PIPE RESULTS

Descriptive Statistics: test -evaluation ap for bbox, test -evaluation for segm ap

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.7, Batch = 4, ROI = 64, Images per class = 1500, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	84.124	0.106	0.237	83.963	83.975	84.014
test -evaluation for segm ap	15	5	0	83.147	0.126	0.282	82.833	82.942	83.082
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	84.328		84.533					
test -evaluation for segm ap	15	83.385		83.600					

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.7, Batch = 8, ROI = 512, Images per class = 1500, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	78.517	0.0380	0.0849	78.397	78.447	78.522
test -evaluation for segm ap	5	5	0	77.817	0.0537	0.120	77.648	77.732	77.818
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	78.585		78.634					
test -evaluation for segm ap	5	77.903		77.987					

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.9, Batch = 4, ROI = 64, Images per class = 1000, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	89.141	0.106	0.237	88.771	88.913	89.209
test -evaluation for segm ap	5	5	0	89.226	0.180	0.403	88.809	88.851	89.133
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	89.335		89.358					
test -evaluation for segm ap	5	89.646		89.682					

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.9, Batch = 8, ROI = 512, Images per class = 1000, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	90.708	0.0709	0.158	90.512	90.579	90.649
test -evaluation for segm ap	15	5	0	90.478	0.102	0.229	90.266	90.299	90.350
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	90.868		90.909					
test -evaluation for segm ap	15	90.721		90.786					

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.7, Batch = 4, ROI = 512, Images per class = 1500, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	91.654	0.0556	0.124	91.503	91.530	91.695
test -evaluation for segm ap	15	5	0	91.265	0.0508	0.114	91.122	91.167	91.267
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	91.757		91.814					
test -evaluation for segm ap	15	91.363		91.430					

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.7, Batch = 8, ROI = 64, Images per class = 1500, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	89.826	0.0303	0.0678	89.756	89.766	89.800
test -evaluation for segm ap	5	5	0	89.563	0.0796	0.178	89.397	89.423	89.496
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	89.898		89.903					
test -evaluation for segm ap	5	89.738		89.839					

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.9, Batch = 4, ROI = 512, Images per class = 1000, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	80.798	0.0107	0.0240	80.767	80.775	80.798
test -evaluation for segm ap	5	5	0	79.916	0.0380	0.0849	79.841	79.842	79.916
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	80.820		80.828					
test -evaluation for segm ap	5	79.991		80.048					

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.9, Batch = 8, ROI = 64, Images per class = 1000, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	82.110	0.136	0.304	81.779	81.875	82.057
test -evaluation for segm ap	15	5	0	81.767	0.140	0.313	81.394	81.501	81.749
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	82.372		82.596					
test -evaluation for segm ap	15	82.043		82.238					

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.7, Batch = 4, ROI = 512, Images per class =

1000, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	81.804	0.266	0.595	81.228	81.277	81.784
test -evaluation for segm ap	15	5	0	81.974	0.232	0.518	81.510	81.569	81.757
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	82.342		82.713					
test -evaluation for segm ap	15	82.487		82.774					

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.7, Batch = 8, ROI = 64, Images per class = 1000, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	71.413	0.242	0.540	70.562	70.904	71.588
test -evaluation for segm ap	5	5	0	75.111	0.231	0.516	74.269	74.635	75.323
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	71.836		71.952					
test -evaluation for segm ap	5	75.482		75.588					

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.9, Batch = 4, ROI = 512, Images per class = 1500, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	89.810	0.112	0.250	89.479	89.582	89.785
test -evaluation for segm ap	5	5	0	90.735	0.0905	0.202	90.519	90.589	90.689
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	90.049		90.122					
test -evaluation for segm ap	5	90.904		91.065					

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.9, Batch = 8, ROI = 64, Images per class = 1500, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	89.095	0.201	0.450	88.625	88.677	88.992
test -evaluation for segm ap	15	5	0	90.312	0.159	0.355	89.813	89.992	90.313
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	89.566		89.585					
test -evaluation for segm ap	15	90.631		90.759					

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.7, Batch = 4, ROI = 64, Images per class = 1000, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	87.903	0.0872	0.195	87.799	87.799	87.805
test -evaluation for segm ap	15	5	0	89.387	0.0660	0.147	89.231	89.231	89.442
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	88.056		88.249					
test -evaluation for segm ap	15	89.516		89.550					

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.7, Batch = 8, ROI = 512, Images per class = 1000, Num Classes = 5, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	90.894	0.0781	0.175	90.716	90.722	90.889
test -evaluation for segm ap	5	5	0	91.000	0.0799	0.179	90.802	90.830	90.993
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	91.069		91.097					
test -evaluation for segm ap	5	91.175		91.233					

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.9, Batch = 4, ROI = 64, Images per class = 1500, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	5	5	0	72.368	0.343	0.767	71.519	71.542	72.727
test -evaluation for segm ap	5	5	0	76.251	0.320	0.717	75.452	75.471	76.681
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	5	73.015		73.124					
test -evaluation for segm ap	5	76.815		76.906					

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.9, Batch = 8, ROI = 512, Images per class = 1500, Num Classes = 10, Type Dataset = Machine parts

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median
test -evaluation ap for bbox	15	5	0	86.303	0.102	0.229	86.092	86.129	86.287
test -evaluation for segm ap	15	5	0	85.453	0.0791	0.177	85.292	85.301	85.438
Variable	Epochs	Q3		Maximum					
test -evaluation ap for bbox	15	86.486		86.684					
test -evaluation for segm ap	15	85.611		85.731					

One-Way ANOVA Bounding Box vs. Experiments Legos Dataset

LEGO RESULTS

Analysis of Means

* NOTE * Cannot draw the interval plot for the Games-Howell procedure. Interval plots for comparisons are illegible with more than 45 intervals.

Method

Null hypothesis All means are equal
Alternative hypothesis Not all means are equal
Significance level $\alpha = 0.05$

Equal variances were not assumed for the analysis.

Factor Information

Factor	Levels	Values
num iters	16	3120, 4680, 6247, 6248, 9367, 9368, 9372, 12500, 14056, 18744, 18747, 18748, 28120, 28124, 37500, 56248

Welch's Test

Source	DF Num	DF Den	F-Value	P-Value
num iters	15	23.0293	478.63	0.000

Model Summary

R-sq	R-sq(adj)	R-sq(pred)
99.08%	98.86%	98.65%

Means

num iters	N	Mean	StDev	95% CI
3120	5	80.462	0.283	(80.110, 80.814)
4680	5	84.7071	0.0718	(84.6180, 84.7962)
6247	5	84.019	0.319	(83.622, 84.415)
6248	5	83.783	0.251	(83.471, 84.095)
9367	3	83.3655	0.1505	(82.9917, 83.7393)
9368	7	82.813	0.877	(82.002, 83.624)
9372	5	73.921	0.393	(73.433, 74.409)
12500	5	83.4874	0.1829	(83.2603, 83.7145)
14056	5	87.5217	0.1395	(87.3486, 87.6949)
18744	5	84.4207	0.1444	(84.2414, 84.5999)
18747	5	86.328	0.339	(85.907, 86.749)
18748	5	85.122	0.312	(84.734, 85.510)
28120	5	85.554	0.319	(85.157, 85.950)
28124	5	85.836	0.291	(85.474, 86.198)
37500	5	76.762	0.371	(76.301, 77.222)
56248	5	85.4434	0.1661	(85.2371, 85.6496)

Games-Howell Pairwise Comparisons

Grouping Information Using the Games-Howell Method and 95% Confidence

num iters	N	Mean	Grouping
14056	5	87.5217	A
18747	5	86.328	B
28124	5	85.836	B C
28120	5	85.554	B C D
56248	5	85.4434	C
18748	5	85.122	C D E
4680	5	84.7071	D E F
18744	5	84.4207	E F G
6247	5	84.019	F G H
6248	5	83.783	G H
12500	5	83.4874	H
9367	3	83.3655	H
9368	7	82.813	G H
3120	5	80.462	I
37500	5	76.762	J
9372	5	73.921	K

Means that do not share a letter are significantly different.

Descriptive Statics Bounding Box and Segmentation AP Legos Dataset

LEGO RESULTS

All trial means

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.7, Batch = 4, ROI = 512, Images per class = 1000, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	84.019	0.143	0.319	0.102	83.561	83.781
test -evaluation for segm ap	5	5	0	81.314	0.165	0.369	0.136	80.842	80.987
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	5	84.004		84.264	84.460	0.900			
test -evaluation for segm ap	5	81.225		81.685	81.739	0.897			

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.7, Batch = 8, ROI = 64, Images per class = 1000, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	83.638	0.181	0.405	0.164	83.205	83.296
test -evaluation for segm ap	15	5	0	80.723	0.205	0.459	0.210	80.201	80.362
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	15	83.504		84.046	84.210	1.005			
test -evaluation for segm ap	15	80.574		81.160	81.410	1.209			

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.9, Batch = 4, ROI = 512, Images per class = 1500, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	85.443	0.0743	0.166	0.0276	85.235	85.265
test -evaluation for segm ap	15	5	0	83.405	0.0368	0.0823	0.0068	83.275	83.335
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	15	85.524		85.582	85.591	0.356			
test -evaluation for segm ap	15	83.416		83.469	83.498	0.224			

Results for LR = 0.001, LRD = 0.01, LRDdecline = 0.9, Batch = 8, ROI = 64, Images per class = 1500, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	82.320	0.123	0.275	0.075	81.929	82.043
test -evaluation for segm ap	5	5	0	80.705	0.125	0.280	0.078	80.326	80.423
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	5	82.408		82.553	82.606	0.677			
test -evaluation for segm ap	5	80.799		80.939	81.032	0.706			

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.7, Batch = 4, ROI = 64, Images per class = 1000, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	83.487	0.0818	0.183	0.0334	83.269	83.309
test -evaluation for segm ap	5	5	0	81.770	0.0712	0.159	0.0253	81.526	81.611
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	5	83.536		83.641	83.735	0.466			
test -evaluation for segm ap	5	81.849		81.889	81.920	0.394			

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.7, Batch = 8, ROI = 512, Images per class = 1000, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	84.421	0.0646	0.144	0.0208	84.171	84.307
test -evaluation for segm ap	15	5	0	82.371	0.0643	0.144	0.0207	82.167	82.233
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	15	84.465		84.512	84.540	0.370			
test -evaluation for segm ap	15	82.381		82.504	82.519	0.352			

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.9, Batch = 4, ROI = 64, Images per class = 1500, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	85.836	0.130	0.291	0.085	85.420	85.544
test -evaluation for segm ap	15	5	0	82.428	0.129	0.288	0.083	82.061	82.178
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	15	85.903		86.095	86.097	0.678			
test -evaluation for segm ap	15	82.343		82.721	82.726	0.666			

Results for LR = 0.001, LRD = 0.10, LRDdecline = 0.9, Batch = 8, ROI = 512, Images per class = 1500, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	84.707	0.0321	0.0718	0.0051	84.601	84.647
test -evaluation for segm ap	5	5	0	81.697	0.0461	0.103	0.0106	81.640	81.641
Variable	Epochs	Median		Q3	Maximum	Range			
test -evaluation ap for bbox	5	84.698		84.772	84.789	0.188			
test -evaluation for segm ap	5	81.644		81.779	81.879	0.239			

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.7, Batch = 4, ROI = 64, Images per class = 1500, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	73.921	0.176	0.393	0.154	73.459	73.618
test -evaluation for segm ap	5	5	0	77.201	0.171	0.382	0.146	76.720	76.811
Variable	Epochs	Median	Q3	Maximum	Range				
test -evaluation ap for bbox	5	73.877	74.245	74.538	1.079				
test -evaluation for segm ap	5	77.293	77.546	77.639	0.919				

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.7, Batch = 8, ROI = 512, Images per class = 1500, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	87.522	0.0624	0.139	0.0194	87.366	87.391
test -evaluation for segm ap	15	5	0	84.575	0.0797	0.178	0.0318	84.414	84.434
Variable	Epochs	Median	Q3	Maximum	Range				
test -evaluation ap for bbox	15	87.551	87.637	87.724	0.357				
test -evaluation for segm ap	15	84.569	84.719	84.869	0.455				

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.9, Batch = 4, ROI = 64, Images per class = 1000, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	76.762	0.166	0.371	0.138	76.365	76.429
test -evaluation for segm ap	15	5	0	79.547	0.0635	0.142	0.0202	79.325	79.437
Variable	Epochs	Median	Q3	Maximum	Range				
test -evaluation ap for bbox	15	76.690	77.129	77.281	0.917				
test -evaluation for segm ap	15	79.555	79.653	79.718	0.394				

Results for LR = 0.025, LRD = 0.01, LRDdecline = 0.9, Batch = 8, ROI = 512, Images per class = 1000, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	83.783	0.112	0.251	0.063	83.385	83.544
test -evaluation for segm ap	5	5	0	82.961	0.121	0.272	0.074	82.511	82.715
Variable	Epochs	Median	Q3	Maximum	Range				
test -evaluation ap for bbox	5	83.892	83.966	84.032	0.647				
test -evaluation for segm ap	5	83.041	83.166	83.177	0.666				

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.7, Batch = 4, ROI = 512, Images per class = 1500, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
----------	--------	---	----	------	---------	-------	----------	---------	----

test -evaluation ap for bbox	5	5	0	85.122	0.140	0.312	0.098	84.693	84.849
test -evaluation for segm ap	5	5	0	83.962	0.0936	0.209	0.0438	83.670	83.788
Variable	Epochs			Median	Q3	Maximum	Range		
test -evaluation ap for bbox	5			85.100	85.405	85.532	0.838		
test -evaluation for segm ap	5			83.989	84.124	84.254	0.584		

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.7, Batch = 8, ROI = 64, Images per class = 1500, Num Classes = 10, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	85.554	0.143	0.319	0.102	85.242	85.287
test -evaluation for segm ap	15	5	0	84.218	0.110	0.246	0.060	84.006	84.038
Variable	Epochs			Median	Q3	Maximum	Range		
test -evaluation ap for bbox	15			85.472	85.861	86.043	0.802		
test -evaluation for segm ap	15			84.091	84.460	84.604	0.599		

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.9, Batch = 4, ROI = 512, Images per class = 1000, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	15	5	0	86.328	0.152	0.339	0.115	85.842	85.986
test -evaluation for segm ap	15	5	0	83.630	0.132	0.296	0.088	83.232	83.361
Variable	Epochs			Median	Q3	Maximum	Range		
test -evaluation ap for bbox	15			86.431	86.618	86.666	0.825		
test -evaluation for segm ap	15			83.587	83.921	83.948	0.717		

Results for LR = 0.025, LRD = 0.10, LRDdecline = 0.9, Batch = 8, ROI = 64, Images per class = 1000, Num Classes = 5, Type Dataset = Lego

Statistics

Variable	Epochs	N	N*	Mean	SE Mean	StDev	Variance	Minimum	Q1
test -evaluation ap for bbox	5	5	0	80.462	0.127	0.283	0.080	80.064	80.173
test -evaluation for segm ap	5	5	0	79.763	0.0746	0.167	0.0278	79.560	79.589
Variable	Epochs			Median	Q3	Maximum	Range		
test -evaluation ap for bbox	5			80.543	80.710	80.728	0.664		
test -evaluation for segm ap	5			79.810	79.914	79.938	0.379		

One-Way ANOVA Bounding Box vs. Experiments All Data

FULL EXPERIMENT DESIGN

One-way ANOVA: test -evaluation ap for bbox_1 versus Column2

* NOTE * Cannot draw the interval plot for the Games-Howell procedure. Interval plots for comparisons are illegible with more than 45 intervals.

Method

Null hypothesis All means are equal
Alternative hypothesis Not all means are equal
Significance level $\alpha = 0.05$
Rows unused 6

Equal variances were not assumed for the analysis.

Factor Information

Factor	Levels	Values
Column2	32	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

Welch's Test

Source	DF Num	DF Den	F-Value	P-Value
Column2	31	44.8772	4096.22	0.000

Model Summary

R-sq	R-sq(adj)	R-sq(pred)
99.70%	99.62%	99.53%

Means

Column2	N	Mean	StDev	95% CI
1	5	89.141	0.237	(88.846, 89.436)
2	5	83.4874	0.1829	(83.2603, 83.7145)
3	5	82.320	0.275	(81.979, 82.661)
4	5	89.8258	0.0678	(89.7416, 89.9100)
5	5	84.019	0.319	(83.622, 84.415)
6	5	80.7979	0.0240	(80.7681, 80.8277)
7	5	78.5171	0.0849	(78.4117, 78.6225)
8	5	84.7071	0.0718	(84.6180, 84.7962)
9	5	84.124	0.237	(83.830, 84.418)
10	5	85.836	0.291	(85.474, 86.198)
11	5	83.638	0.405	(83.135, 84.141)
12	5	82.110	0.304	(81.733, 82.488)
13	5	85.4434	0.1661	(85.2371, 85.6496)
14	5	91.6538	0.1242	(91.4996, 91.8081)
15	5	90.7085	0.1585	(90.5117, 90.9053)
16	5	84.4207	0.1444	(84.2414, 84.5999)
17	5	73.921	0.393	(73.433, 74.409)
18	5	72.368	0.767	(71.415, 73.321)
19	5	71.413	0.540	(70.743, 72.084)
20	5	80.462	0.283	(80.110, 80.814)

21	5	89.810	0.250	(89.499, 90.121)
22	5	85.122	0.312	(84.734, 85.510)
23	5	83.783	0.251	(83.471, 84.095)
24	5	90.8944	0.1747	(90.6775, 91.1114)
25	5	76.762	0.371	(76.301, 77.222)
26	5	87.9032	0.1950	(87.6612, 88.1453)
27	5	89.095	0.450	(88.537, 89.654)
28	5	85.554	0.319	(85.157, 85.950)
29	5	81.804	0.595	(81.066, 82.543)
30	5	86.328	0.339	(85.907, 86.749)
31	5	87.5217	0.1395	(87.3486, 87.6949)
32	5	86.303	0.229	(86.020, 86.587)

Games-Howell Pairwise Comparisons

Grouping Information Using the Games-Howell Method and 95% Confidence

Column2	N	Mean	Grouping
14	5	91.6538	A
24	5	90.8944	B
15	5	90.7085	B
4	5	89.8258	C
21	5	89.810	C
1	5	89.141	C
27	5	89.095	C D
26	5	87.9032	D E
31	5	87.5217	E
30	5	86.328	F
32	5	86.303	F
10	5	85.836	F G
28	5	85.554	F G H
13	5	85.4434	F G H
22	5	85.122	G H I
8	5	84.7071	H I J
16	5	84.4207	I J K
9	5	84.124	J K L
5	5	84.019	J K L
23	5	83.783	K L
11	5	83.638	J K L
2	5	83.4874	L M
3	5	82.320	N
12	5	82.110	N
29	5	81.804	M N O
6	5	80.7979	O
20	5	80.462	O
7	5	78.5171	P
25	5	76.762	Q
17	5	73.921	R
18	5	72.368	R S
19	5	71.413	S

Means that do not share a letter are significantly different.

BIBLIOGRAPHY

- “7 best image recognition apis: Nordic apis —.” [Online]. Available: <https://nordicapis.com/7-best-image-recognition-apis/>,journal={NordicAPIs},publisher={NordicAPIsAB},author={Simpson,J.},year={2020},month={Apr}
- “Computer vision market,” 2018. [Online]. Available: <https://www.marketsandmarkets.com/PressReleases/computer-vision.asp>
- “Covid-19 outlook for the us technology industry,” Nov 2020. [Online]. Available: <https://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/covid-19-outlook-on-technology-industry.html>
- “Enterprise ai,” Oct 2020. [Online]. Available: <https://c3.ai/>
- “Math types utilities (mathutils),” Nov 2020. [Online]. Available: <https://docs.blender.org/api/current/mathutils.html>
- “Object recognition,” 2020. [Online]. Available: <https://library.vuforia.com/articles/Training/Object-Recognition>
- “The state of automation, artificial intelligence and machine learning in network management,” 2020.
- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: <http://tensorflow.org/>
- Bobriakov, I., “Comparison of top 6 cloud apis for computer vision,” Oct 2018. [Online]. Available: <https://medium.com/activewizards-machine-learning-company/comparison-of-top-6-cloud-apis-for-computer-vision-ebf2d299be73>
- Brownlee, J., *Better Deep Learning Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

- Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y., “Deformable convolutional networks,” *CoRR*, vol. abs/1703.06211, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06211>
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., and Katam, H., “Blenderproc,” *CoRR*, vol. abs/1911.01911, 2019. [Online]. Available: <http://arxiv.org/abs/1911.01911>
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A., “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, June 2010.
- Georgiou, T., Liu, Y., Chen, W., and Lew, M. S., “A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision,” *Int. J. Multim. Inf. Retr.*, vol. 9, no. 3, pp. 135–170, 2020. [Online]. Available: <https://doi.org/10.1007/s13735-019-00183-w>
- Girshick, R. B., “Fast R-CNN,” *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: <http://arxiv.org/abs/1504.08083>
- Girshick, R. B., Donahue, J., Darrell, T., and Malik, J., “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: <http://arxiv.org/abs/1311.2524>
- Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S., “Deep learning for visual understanding,” *Neurocomput.*, vol. 187, no. C, p. 27–48, Apr. 2016. [Online]. Available: <https://doi.org/10.1016/j.neucom.2015.09.116>
- He, K., Gkioxari, G., Dollár, P., and Girshick, R. B., “Mask R-CNN,” *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06870>
- He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K., “On pre-trained image features and synthetic images for deep learning,” 2017.
- Hodan, T., Sundermeyer, M., Drost, B., Labbe, Y., Brachmann, E., Michel, F., Rother, C., and Matas, J., “Bop challenge 2020 on 6d object localization,” 2020.
- Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K., “Speed/accuracy trade-offs for modern convolutional object detectors,”

- CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: <http://arxiv.org/abs/1611.10012>
- JORDAN, J., “Evaluating image segmentation models.” Dec 2018. [Online]. Available: <https://www.jeremyjordan.me/evaluating-image-segmentation-models/>
- Jordan, M. I. and Mitchell, T. M., “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015. [Online]. Available: <https://science.sciencemag.org/content/349/6245/255>
- Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P., “Panoptic segmentation,” 2019.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: <https://doi.org/10.1145/3065386>
- Li, Z. and Hoiem, D., “Learning without forgetting,” *CoRR*, vol. abs/1606.09282, 2016. [Online]. Available: <http://arxiv.org/abs/1606.09282>
- Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: <http://arxiv.org/abs/1405.0312>
- Paleyes, A., Urma, R.-G., and Lawrence, N. D., “Challenges in deploying machine learning: a survey of case studies,” 2020.
- Peng, X., Sun, B., Ali, K., and Saenko, K., “Exploring invariances in deep convolutional neural networks using synthetic images,” *CoRR*, vol. abs/1412.7122, 2014. [Online]. Available: <http://arxiv.org/abs/1412.7122>
- Peng, X., Sun, B., Ali, K., and Saenko, K., “Learning deep object detectors from 3d models,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV ’15. USA: IEEE Computer Society, 2015, p. 1278–1286. [Online]. Available: <https://doi.org/10.1109/ICCV.2015.151>
- Planche, B., Wu, Z., Ma, K., Sun, S., Kluckner, S., Chen, T., Hutter, A., Zakharov, S., Kosch, H., and Ernst, J., “Depthsynth: Real-time realistic synthetic data generation from CAD models for 2.5d recognition,” *CoRR*, vol. abs/1702.08558, 2017. [Online]. Available: <http://arxiv.org/abs/1702.08558>
- Rajpura, P. S., Hegde, R. S., and Bojinov, H., “Object detection using deep cnns trained on synthetic images,” *CoRR*, vol. abs/1706.06782, 2017. [Online]. Available: <http://arxiv.org/abs/1706.06782>

- Ren, S., He, K., Girshick, R. B., and Sun, J., “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: <http://arxiv.org/abs/1506.01497>
- Richter, S., Vineet, V., Roth, S., and Koltun, V., “Playing for data: Ground truth from computer games,” 08 2016.
- Ryan, B., Joiner, B. L., and Cryer, J.
- Schraml, D., “Physically based synthetic image generation for machine learning: a review of pertinent literature,” in *Other Conferences*, 2019.
- Simonyan, K. and Zisserman, A., “Very deep convolutional networks for large-scale image recognition,” *arXiv 1409.1556*, 09 2014.
- Spisak, J., 2018. [Online]. Available: <https://developers.facebook.com/blog/post/2018/05/02/onnx-expansion-speeds-research-production/>
- Su, H., Qi, C. R., Li, Y., and Guibas, L. J., “Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views,” *CoRR*, vol. abs/1505.05641, 2015. [Online]. Available: <http://arxiv.org/abs/1505.05641>
- Sundermeyer, M., Marton, Z., Durner, M., Brucker, M., and Triebel, R., “Implicit 3d orientation learning for 6d object detection from RGB images,” *CoRR*, vol. abs/1902.01275, 2019. [Online]. Available: <http://arxiv.org/abs/1902.01275>
- Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., “Domain randomization for transferring deep neural networks from simulation to the real world,” *CoRR*, vol. abs/1703.06907, 2017. [Online]. Available: <http://arxiv.org/abs/1703.06907>
- Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Bochoon, S., and Birchfield, S., “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” 2018.
- Wang, Y., Wei, G., and Brooks, D., “Benchmarking tpu, gpu, and CPU platforms for deep learning,” *CoRR*, vol. abs/1907.10701, 2019. [Online]. Available: <http://arxiv.org/abs/1907.10701>
- Wu, X., Sahoo, D., and Hoi, S. C. H., “Recent advances in deep learning for object detection,” *Neurocomputing*, vol. 396, pp. 39–64, 2020. [Online]. Available: <https://doi.org/10.1016/j.neucom.2020.01.085>
- Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R., “Detectron2,” <https://github.com/facebookresearch/detectron2>, 2019.
- Wu, Z., Song, S., Khosla, A., Tang, X., and Xiao, J., “3d shapenets for 2.5d object recognition and next-best-view prediction,” *CoRR*, vol. abs/1406.5670, 2014. [Online]. Available: <http://arxiv.org/abs/1406.5670>

- Yan, W., “Augmented reality applied to LEGO construction: Ar-based building instructions with high accuracy & precision and realistic object-hand occlusions,” *CoRR*, vol. abs/1907.12549, 2019. [Online]. Available: <http://arxiv.org/abs/1907.12549>
- Yang, Z., Wang, Y., Liu, C., Chen, H., Xu, C., Shi, B., Xu, C., and Xu, C., “LegoNet: Efficient convolutional neural networks with lego filters,” ser. *Proceedings of Machine Learning Research*, Chaudhuri, K. and Salakhutdinov, R., Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 7005–7014. [Online]. Available: <http://proceedings.mlr.press/v97/yang19c.html>
- Yin, X., Chen, Y., Bouferguene, A., Zaman, H., Al-Hussein, M., and Kurach, L., “A deep learning-based framework for an automated defect detection system for sewer pipes,” *Automation in Construction*, vol. 109, p. 102967, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0926580519307411>
- Zou, Z., Shi, Z., Guo, Y., and Ye, J., “Object detection in 20 years: A survey,” *CoRR*, vol. abs/1905.05055, 2019. [Online]. Available: <http://arxiv.org/abs/1905.05055>
- Židek, K., Lazorík, P., Pitel, J., and Hošovský, A., “An automated training of deep learning networks by 3d virtual models for object recognition,” *Symmetry*, vol. 11, no. 4, p. 496, Apr 2019. [Online]. Available: <http://dx.doi.org/10.3390/sym11040496>