SMU Data Science Review

Volume 2 | Number 3 Article 11

2019

Mapping Relationships and Positions of Objects in Images Using Mask and Bounding Box Data

Jaime M. Villanueva Jr
Southern Methodist University, jvillanueva@smu.edu

Anantharam Subramanian
Southern Methodist University, anantharams@mail.smu.edu

Vishal Ahir
Southern Methodist University, vahir@mail.smu.edu

Andrew Pollock
Getty Images, andrew.pollock@gettyimages.com

Mapping Relationships and Positions of Objects in Images Using Mask and Bounding Box Data

Anantharam Subramanian[1], Vishal Ahir[1], Jaime Villanueva[1], and Andrew Pollock[2]

[1] Masters of Science in Data Science, Southern Methodist University, Dallas, TX 75275 USA
{anantharams,vahir, jvillanueva}@smu.edu
[2] Getty Images, Seattle, WA 98104
Andrew.Pollock@gettyimages.com

Abstract. In this paper we present novel methods for automatically annotating images with relationship and position tags that are derived using mask and bounding box data. A Mask Region-based Convolutional Neural Network (Mask R-CNN) is used as the foundation for the object detection process. The relationships are found by manipulating the bounding box and mask segmentation outputs of a Mask R-CNN. The absolute positions, the positions of the objects relative to the image, and the relative positions, the positions of objects relative to the other objects, are then associated with the images as annotations that are output in order to assist with the retrieval of those images with keyword searches. Programs were developed in python to perform the image analysis and as well as streamline the manual annotation of the images for testing. Image annotations that specify the relative location of objects to each other in an image would empower more nuanced searches allowing for finer filtering of images that contain common objects. Because the masks are manipulated as boolean matrices, the processing is fast. Also, this approach is model agnostic being able to work with any model that outputs a boolean mask and bounding box data.

1 Introduction

Computer algorithms that can detect and classify objects in an image have become very sophisticated to the point that in certain venues, they have been able to outperform humans.[1] Even though the development of current algorithms continues to be able to classify an increasing assortment of objects, the need to be able to extract even more information from digital images continues to push this development of computer models. One of these needs is the ability to not only classify objects in an image, but also to detect the relationship of these

[1] Parthasarathy, Dhruv. "A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN." Medium, 27 Apr. 2017, https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4.

objects to each other. Current computer models can group pixels together in order to delineate and classify objects, but there is no effective way to express the location of the objects in relation to each other within in image.

There is a need for an automated way to analyze images and classify not only what objects are in the image, but also the position of those objects with reference to both the image and to the other objects in the image. To address this, a program is developed that will output annotations to a python dictionary that matches the annotations to an image reference. In this way, the image can be located by using keywords that can be matched against the program outputs in the dictionary. The absolute position is recorded in two ways. First, in a standard 3 x 3 grid using 'top', 'center', 'bottom', 'left', and 'right'. This grid is flexible and can be moved around to accommodate different filtering needs. Second, the image can be divided up into a variable number of sections labeled with letters and numbers along each respective axis. The relative position will output the relationships, 'next to', 'above', 'below', 'touching', 'on', and 'in'. Being able to annotate both the absolute and relative positions enables a user to differentiate between photos that may have many of the same objects.

The need for this ability came about as a natural result of the progress in the field of Computer vision (CV). This field involves extracting information from images that can be useful for understanding the context of the image. This process is concerned with using algorithms to automatically process the images and generate this information. Earlier studies from several decades ago are stepping stones for many computer vision algorithms today. Digital image processing was a similar field, however, gaining complete scene understanding was never the end goal in that field. In contrast, today there are many fields in digital image processing which try to understand the objects within an image as well as the scene context. A few of these are: image processing and analysis, machine vision, imaging, and pattern recognition.

Other common CV applications include inspection of objects at any stage on a production line. For e.g. identifying apples that are bruising, checking that bottles are filled completely with beverages, etc. When used indoors, lighting is seldom an issue but CV is also as widely used outdoors. One common use being reading registration plates on tollbooths at any given time or conditions like rain etc. With progress made in deep learning, we now have models that can generate 3D shapes of objects, from images and/or maps, efficiently. This ability has applications within disciplines like robotics, artificial intelligence, and medical imaging.

The progress in digital image process technologies was accompanied by an increasing amount of digital images to process. The larger any one collection of images becomes, the more structured the organization has to be in order to be able to find a particular image when needed. To aid in this process, images are tagged with keywords that describe the objects, actions of objects, or the scene context in an image. At first, the tagging process was performed manually, but machine learning models were developed that are capable of automatically

generating annotations for images. These auto-generated tags aid in the efficient retrieval of images from image databases.

How specifically a search function can retrieve an image based on search keywords is important for any organization that digitizes, stores, and catalogs large amounts of images. The ability of the search algorithm to weed out unrelated or undesired images is dependent on the quality of the tags. This quality can be enhanced by descriptions of not only objects, actions, or scene context, but also by including words that describe the relationships between objects. A program that can auto-generate words that accurately describe this kind of relationship between objects in an image will provide more specific search results.

These words often grammatically take the forms of prepositional phrases that are objects of verbs, and grouped together these can be described as predicates that describe the positioning of noun objects in reference to each other. Examples of these predicates are 'standing on top of', or 'hiding behind the', or 'sitting inside of'. Determining how well these predicates were generated is ultimately judged by the accuracy of a keyword search. For example, if the search is 'man sitting in car', then the images returned should not show a man next to a car, or behind a car, or on top of a car.

Part of the challenge in relating objects in an image spatially has to do with how neural networks process images. Individual image pixels are read into a big array which an algorithm segments as part of the object detection process. "But because categorization neural network want to reduce a large amount of pixels to a small amount of data ... they also lose the ability to precisely localize object instances..." [2] Though this fact affects the precision with which the edges of pixels can be located, a Convolutional neural network (CNN) still has the ability to determine the relative positions of objects to each other.

The pre-trained model, Mask Region-based Convolutional Neural Network (Mask R-CNN), is used to aid in the detection and classification process. The model weights being used are weights that were trained on Microsoft's Common Obects in Context (MS-COCO) dataset. This is a dataset that was developed by Microsoft to further the research in the field of CV. It contains over 2 million labeled instances with over 300,00 labeled images [6]. The weights and the models are all used in the Keras environment. Keras is a popular open-source neural net library that offers resources for those wanting to train a neural net. [3]The computational backend used in Keras is Tensorflow which is a library that allows for efficient computations of complex matrix calculations. Keras resources used include VGG16, which is another model that is used for developing neu-

[2] Culurciello, E., "Segmenting, Localizing and Counting Object Instances in an Image." Towards Data Science, 26 Oct. 2017, https://towardsdatascience.com/segmenting-localizing-and-counting-object-instances-in-an-image-878805fef7fc

[3] Heller, Martin. "What Is Keras? The Deep Neural Network API Explained." InfoWorld, 28 Jan. 2019, https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html

ral networks.[4] This model was used initially when learning about Convolutional Neural Networks (CNN), but it was eventually replaced with Mask R-CNN.

Following the introduction are discussions of Tensorflow and Artificial Neural Networks. Then the results of the program are presented followed by an analysis of these results. Before the conclusion, ethical concerns associated with this program are discussed.

2 TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.[5] Google designed TensorFlow to run on variety of platforms like GPU, CPU and TPU and from desktops, server clusters, mobile and edge devices

2.1 Computation Graphs

By creating and computing operations that can interact with each other, TensorFlow allows us to implement Machine Learning algorithms. Such interactions form what is referred to as a "computation graph". Computation graph, for example, is a set of interconnected entities called nodes/vertices. These nodes are connected to each other via edges. The edges allow data to flow from one entity to another. In TensorFlow, every node is considered as an operation (add, subtract, multiply, etc) to be applied on some input that will be received via edges and can generate output that is passed to other nodes. Operations can be anything from simple arithmetic to complex ones. At times they may only create summaries or generate constant values. Such computation graphs can have direct dependency or indirect dependency between nodes. Nodes that relate to same edge have direct dependency whereas if there are multiple edges and 1 or more nodes between 2 nodes, they are considered indirectly dependent. As an example, below, B is directly connected to A and E is indirectly connected to C. Working with TensorFlow involves creating a computation graph and executing it. After the graph is constructed, we need to create a session and run it. A session object is part of TensorFlow API that communicates between Python objects and data at our end and while doing that it also interacts with computational system where memory is allocated for objects we define along with intermediate variables stored. TensorFlow computes only the essential nodes according to set of dependencies. Irrespective of size of graph, only a small portion can be executed as needed. To describe above graph, first we import TensorFlow and perform next steps.A = Constant(4), B = Constant(3), C = Add A and B, D = Multiply A and B, E = Sub result of D and C, final result 5.

[4] VGG16 – Convolutional Network for Classification and Detection, `https://neurohive.io/en/popular-networks/vgg16/`

[5] What Is TensorFlow? Introduction, Architecture & Example. Accessed 17 Sep. 2019. `https://www.guru99.com/what-is-tensorflow.html`

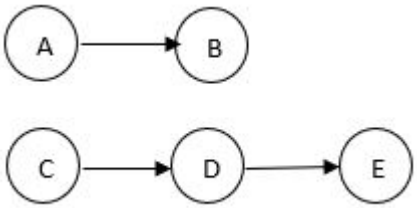

Fig. 1. Direct and Indirect Graphs

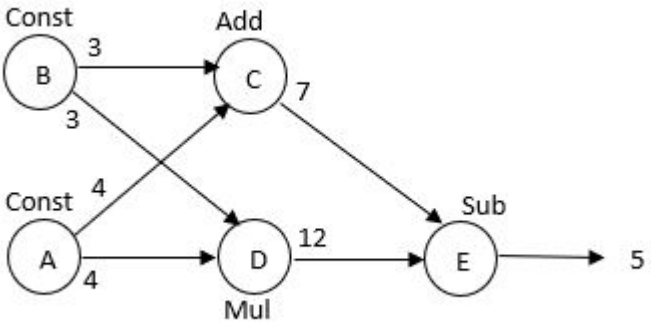

Fig. 2. Sample TensorFlow Graph Execution

2.2 TensorFlow Building Blocks

Variables and placeholders are core building blocks for TensorFlow. Variables don't loose data each time the session is run. They can maintain fixed state in graph. Current state information is important as it will influence how they will change in following iteration. Variables are computed only when the model is run. Two functions getvariables() and Variable() are used to get existing and variables vice-versa. The designated built-in structure for feeding input values are called placeholders. Placeholders are empty variables that will be filled with data later. Placeholders have an argument called shape, in case it is not passed or passed as None, they can be fed with any size of data. At the time when placeholders are defined they must be fed with input values else an exception is thrown.

2.3 Gradient Descent Optimizer

It is always required to have a good measure for evaluating model's performance. Discrepancy between prediction and actual observations is often referred to as a loss function or objective. An optimized model is the one that has right parameters to minimize this difference. A common loss function typically used is MSE or Mean Squared Error, where the average of squared differences or residuals is taken across the samples. In case of categorical data, a common loss function is cross entropy. It is a measure of similarity between two distributions. In case of deep learning classification models, the probabilities of each classification class is the output, such probabilities are compared with true class. Similar distributions have less cross entropy. To minimize loss function, a common approach is gradient descent. Gradient Descent algorithms work on highly complicated network architectures. Alternatively, Stochastic Gradient Descent takes only a subset of data set as input referred to as mini-batches. They do tend to have a high variance. TensorFlow computes gradients on its own by deriving them from previous operations and structure of computation graph.

3 Artificial Neural Networks

In general, artificial neural networks (ANN) are mathematical models inspired by biological neural networks in the brain which can be trained to learn a specific output for a given input. In a brain, information is passed by the firing of collections of neurons, and the different patterns of the grouping of these neurons being fired are associated with specific results. An ANN works similarly in that a particular output of the network can be taught by finding patterns in numbers in the input of the network. The neuron, the basic unit of an ANN, is referred to as a node. The initial layer of nodes holds numerical information representing the input data. The final output is a number which is further used for some function. Often this function is classification. In-between the input and output, there can be additional layers, usually called hidden layers, with each

layer consisting of nodes. The number of nodes in each layer and the number of layers between the input and output can vary depending on the type of ANN being used and the application for this use. Except for the input and output, in a fully connected model, every single node in one layer passes information to every node in the next layer. Conversely, every node in any one layer also receives information from every node in the preceding layer. The information

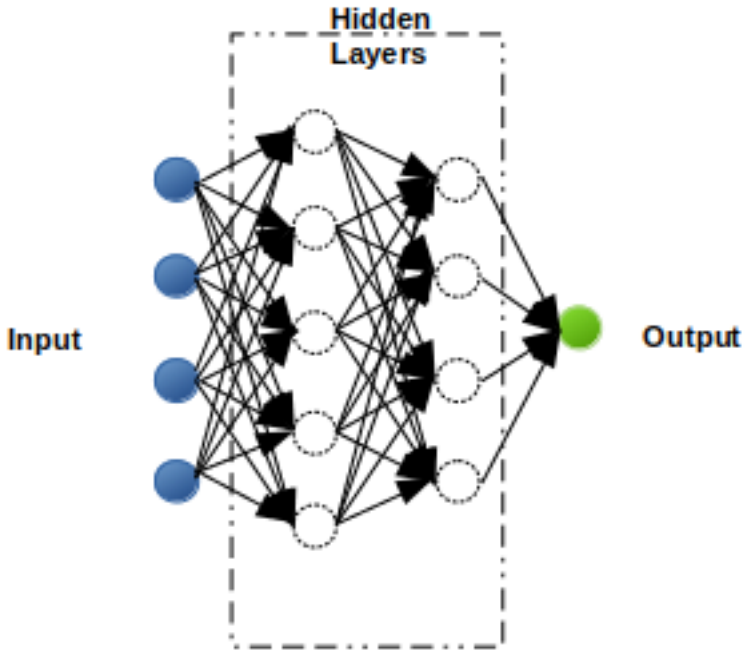

Fig. 3. Artificial Neural Network

being passed is in the form of numbers, and the numbers any one node receives is added to some bias constant. Then this result is fed through a mathematical equation referred to as an activation function. There are many types of activation functions, but typically, the activation function transforms the number into another scaled value. This scaled value becomes the output for that node and affects what value is passed on to all the nodes in the next layer. The number being passed between nodes consists of a weighting coefficient that is multiplied by the value being output from the node. If the node is from the initial input layer, then this value is the numeric representation of the input data. If the node is an internal layer, then this value is the output of the activation function. The steps summarizing the general operation of a neural network can be expressed by saying that first, a list of values each representing a node in the initial input layer each gets multiplied by a weighting coefficient. The result of this multiplication

is sent to every node in the next layer. In each of those nodes, a bias constant is added. Then the result of this addition is fed to an activation function which outputs a scaled value. The process then repeats for the next layer with a new set of values for the weighting coefficients.

The general steps for a neural network's operation describes how numbers flow through the network, but not how the network learns. This is achieved through a process known as backpropagation which is a term that describes the network finding the optimal values for the weighting coefficients that are used at each layer. The method used to find these optimal values is gradient descent.[6] When training a neural network, the desired output is indicated which then forces values for its output node that backpropagate through the entire network. For example, if the output desired is a specific classification from a list of four different classifications, then the value in the node in the output layer which is associated with that class needs to be maximized. Since the value in that node is a result of every node in the preceding layer all of which individually are results of all the nodes in their preceding layer, the process is complex, especially considering that the optimal value of the weighting coefficients has to work for all the classifications, not just one of them. The complexity grows with the size of the input and number of layers added between the input and output.

An example of this complexity can be seen by considering the total number of parameters involved with the processing of a small grayscale image with a low resolution of 20x20. This means 400 pixels resulting in an initial input of 400 nodes each with a normalized value determining the amount of shade in the pixel. Each of these 400 nodes would be multiplied by a weight, then passed to every node in the next layer where it would be added to a bias. From the input, there are already 400 initial values, and 400 weights. If the next layer had a very low number of nodes such as 10, then each input node would send its initial value multiplied by the weight to every node where it would be added to a bias. That means every node in the next layer uses 400 weights, 400 initial values, and 1 bias for a total of 8010 parameters for moving from the initial layer to one hidden layer. A neural network can have several layers with each layer having thousands of nodes. If the input image had a higher resolution such as 100x100, the total number of parameters can very quickly increase.

One of the methods for reducing the number of parameters in the network is in the choice of the activation function. A popular activation function for neural networks in the past has been the same function used by logistic regression, the sigmoid function, which outputs a continuous value between 0 and 1. A more recent activation function that is popular is the ReLu function which stands for rectified linear unit. This function outputs a 0 if the input is less than 0, otherwise it just outputs the input value. When a 0 is output, this effectively nullifies the value and the parameters that would be transferred to the next layer

[6] Ognjanovski, Gavril. "Everything You Need to Know about Neural Networks and Backpropagation — Machine Learning Made Easy. . . ." Medium, 7 Feb. 2019, `https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a`

from this node. This helps to introduce sparsity into the network. Other types of neural networks have been developed for efficient processing of different types of data. For image data, one of these networks is a CNN.

3.1 Convolutional Neural Network

A CNN is good at processing data that share some spatial relationship. The distance between the data and where it is placed in space affects the patterns that are extracted. This distance would be lost in an ANN because the input would be the values of each pixel transformed into one vector, however, the CNN retains the relationship. One reason this type of neural network is popular for processing images because it has features that can reduce the overall amount of parameters in the network. It does this by exploiting a property of images in that usually, the value of one pixel is very close to the value of all the pixels that are surrounding it. Therefore, every individual pixel does not need to be used in the process of extracting the patterns. This can be accomplished by processing a small area of pixels with a filter. The filter is a matrix where each element has a value that is optimized to detect a specific pattern in the image. This filter size is much smaller than the image size being around 3x3. The filter is also referred to as a window. The window is slid over the entire 2D image pixel by pixel. The dot product between the values in the window and the values in the section of image that is being covered is stored as a single value of a grid in the next layer. The amount that the window is moved is referred to as the stride. If the 3x3 filter is slid with a stride of one, meaning one pixel at a time, then each single value in the grid in the next layer will be formed from overlapping 3x3 sections of the input image. This means that the next layer's dimensions are smaller than the input layer. This is the convolution process and is usually the very first layer of a CNN. The values of the filter are initialized to ones and zeros, and the arrangement of the these values in the filter can emphasize or de-emphasize certain patterns in the underlying image.[7] These patterns are reduced to a single number by the dot product operation in the first layer after the input layer. Then this layer is subjected to a pooling layer where a small section, such as a 2x2 or 3x3, non-overlapping portion of the grid is reduced again to typically the largest value in that small section. This process of convolution and pooling can be repeated several times each of which extracts a higher level combination of features than the previous layer. The end result of this is passed to an output grid which is then vectorized and passed to a fully connected layer that generates a final node value. The fully connected layer operates the same as an ANN. The convolution layers can each be composed of many filters that select for different features. The backpropagation process optimizes the values in each filter. Each of these filters is associated with a node, and each node is

[7] Deshpande, Adit. A Beginner's Guide To Understanding Convolutional Neural Networks. Accessed 17 Sep. 2019. `https://adeshpande3.github.io/adeshpande3.github.io/A-Beginners-Guide-To-Understanding-Convolutional-Neural-Networks/`

associated with a function which determines whether that node will be activated or not. The pattern of activation of the nodes through the network is ultimately what makes the decision at the output nodes.

3.2 Region-based Convolutional Neural Network

A regular CNN is good at training images for classification, but modern computer vision needed a way to not just classify an image with a single object, but it needed a way to classify all the objects within an image. This is problematic because the output of a CNN typically would only classify the input to one class. The image could be split up into areas of focus and searched, but this could computationally be an issue because the number of regions generated that would be needed to get a good classification could reduce the efficiency of the detection process. A region-based convolutional neural network (R-CNN) addresses this issue by beginning the process with a selective search method. This method, first, divides the image into regions, then using distance metrics, hierarchically groups these regions together. This grouping process results in different region proposals. These region proposals are then fed to a CNN in order to extract the features. Once these features are extracted, the process of classification for the region is performed by SVM classification. In addition, a bounding box for the region is determined by linear regression. The area delineated by the bounding box is referred to as a a region of interest (ROI).

3.3 Fast and Faster Region-based Convolutional Neural Network

Though, R-CNN was an improvement over regular CNN, it was still too slow for certain real-time computer vision needs. The search method for dividing the image into regions was relatively slow, and each region needed to be fed to the CNN. Fast Region-based convolutional neural networks (Fast R-CNN) addressed the issue of multiple runs through a CNN by feeding the image first to the CNN. Instead of extracting features, the CNN would output a feature map from which the region proposals would be determined by the selective search method. This feature map is also referred to as a region proposal network (RPN). These regions would be pooled and fed to a fully connected layer for classification. The pooling process consists of down-sampling the features in the ROI. This is necessary in order to introduce noise into the process making sure the classification process can better generalize. The Faster Region-based convolutional neural network (Faster R-CNN) works similarly to the Fast R-CNN, but it substitutes the slower selective search method for generating the region proposals with a different faster network.

3.4 Mask Region-based Convolutional Neural Network

The Mask Region-based convolutional neural network (Mask R-CNN), extends functionality of Faster R-CNN by adding the ability to do instance segmentation. Problems in detection and segmentation can be broadly classified into 4

categories namely, classification, localization, object detection and instance segmentation. The segmentation task creates a pixel wise mask for every object that is found in the image. The full network occurs in two stages. The first stage works similarly to Faster R-CNN, and generates the ROI. Then these ROI are fed to a fully connected convolutional network (FCN) to do instance segmentation. The FCN is trained to produce binary masks within the ROI so that the pixels that are true in the mask correspond to the object being detected in the bounding box. The positive regions that are selected in ROI classification are converted into soft masks. The generated masks are low in resolution of about 28x28 pixels. The smaller mask size helps us to keep the mask branch lighter. In this way Mask R-CNN gives image segmentation, classification, and bounding boxes for objects in an image. The two stages are built on the backbone of a Feature Pyramid Network (FPN). The framework that we utilized for retraining model uses ResNet101 with FPN backbone. The purpose of the FPN is to enhance output from backbone model to better represent objects at multiple scales.

4 Design and Solution

4.1 Model Training

A CNN model using VGG16 was initially developed to detect dogs and cats. This exercise helped to gauge the reliability of using the pre-trained VGG16 weights. During the training of the CNN model, around 28,000 images were used. Then a technique called Transfer Learning was used to train this model to detect different objects. Transfer Learning allows someone building a model to use the weights trained on another set of images as a foundation for learning a new set of images. This helped to get better accuracy during model development while only using around 1000 new training images. The results of this can be seen in Figure 4.

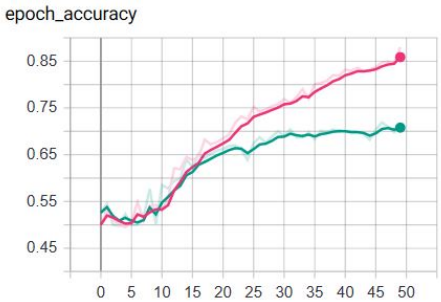

Fig. 4. CNN Transfer Learning Performance

The initial run of the CNN yielded around 94 percent accuracy, and the CNN model developed with Transfer Learning gave a performance on the new set of images of around 72 percent accuracy. The problem with this approach was that even with Transfer Learning, the training time for each new object needed to detect the relationships would yield a model that was limited. Since the object detection ability would hinder our ability to express relationships, other pre-trained models were explored, in particular models were explored that would allow detection of an array of objects commonly found in images allowing us the flexibility to focus on developing the logic for the object relations. The model we ended up utilizing was the Mask R-CNN model developed by the Matterport framework[8]. Even though development started with the new model, preparations were made to cover instances of images that might not be covered by Mask R-CNN, therefore the preparation of validation data sets was started using the VGG annotation tool which was very easy to understand and implement. These sets were focused on annotating situations that the new model was having issues with, specifically, training people inside of cars.

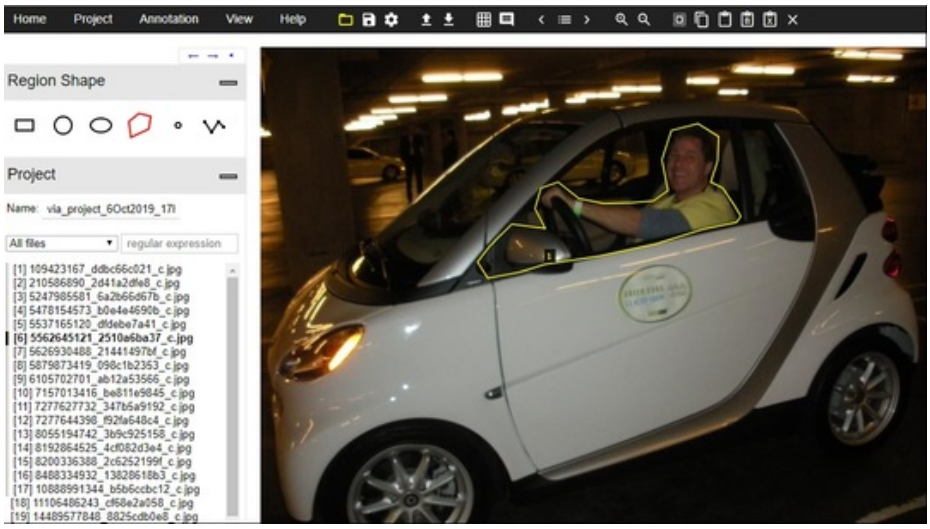

Fig. 5. Annotated Sample

We downloaded 75 images from the Flickr website and split them into train and test sets. Using the VGG annotation tool we created an outline for a person sitting in a car either driving or sitting in backseat. A sample annotated image is shown in figure 5. This tool creates a JSON file with coordinates of the object annotated which are needed for input when training and testing a new model.

[8] Waleed, Abdulla. Matterport/Mask R-CNN. 2017. Matterport, Inc, 2019. `https://github.com/matterport/MaskR-CNN`

Fig. 6. Prediction

4.2 Object Mapping

The main goal with the object mapping is to express a relationship between the objects that can be added to the image annotations so that any image searches can use these annotations for more precise filtering of image banks. The program makes the determinations based on rules. The relationships being defined are a 'next to' relation, an 'above' relation, an 'in' or 'on' relation, a 'below' relation, and a flag for any interaction or touching between objects. The absolute location for each object is also recorded. In this way, a search can specify a desired object in a sector of the image, such as 'cars in sector A2' or 'dogs at the bottom-center' of an image, and the the annotations in the dictionary would be able to reference the appropriate images for retrieval. Also being annotated is relative positions such as 'man next to dog', or 'woman on motorcycle', and likewise these annotations could be used to retrieve appropriate images.

The 'next to' relationship is given if two masks are within a specified tolerance of pixels from each other while checking horizontally. The tolerance number is different for each image category. Because the different objects can have parts like appendages, the above and below relationships are found by using an objects' mass boxes instead of the bounding boxes. For example, the bounding box for the profile of a table would include its legs and be centered in the open space below the table. A mass box will be centered closer to the actual table-top itself.

The mass box for an object is determined by starting with the given coordinates of the bounding box and scanning towards the image one axis at a time, one pixel at a time. At each iteration of pixel movement, the total pixels of the mask remaining in the box is multiplied by the percentage of pixels of the mask remaining in the box. The coordinate will continue iterating all the way to the other end of the box remembering at what value the product was maximal. All four axes are moved in towards the center of the object in this manner.

The 'above' and 'below' tags use the center of the mass box of the object while checking to what degree the pixels from each object aligned with each other. The 'on' and 'in' tags proved troublesome relying on the interaction between an object's pixels and another object's topline to be determined. One other useful relationship that is recorded is an interaction between objects where objects are actually touching. First, each mask is artificially inflated by a specified number of pixels. This is performed by scanning in from all four edges of the image one axis at a time. The scanning looks at the specified number of pixels, and if the first pixel is False, it performs a logical 'OR' operation and replaces all pixels being evaluated by the result. After this is completed for each object's mask, a logical 'AND' is performed between each mask followed by a search for any True values. If any are found, then the objects are determined to be touching.

5 Analysis and Results

A summary of the performance of the three models primarily used can be seen in Table 1. The testing of the COCO weights with our images was expected to be a little higher, but even at 82% accuracy, these weights were very serviceable. The model was tested against the same images that were used to generate the relationship tags.

Table 1. Testing of Model Performance.

Model	Weights	Accuracy
CNN	VGG16	94%
CNN	VGG16 (Transfer Learning)	72%
Mask R-CNN	COCO Mask R-CNN	82%

One interesting thing to note is that the images used for testing the MS-COCO weights as well as the annotation process overwhelmingly had people as the detected object. This spread can be seen in Figure 7. Though, it makes sense that people would normally be a part of images in general, it is noteworthy that many practice runs were run on controlled sets such as dogs, cats, bikes, cars etc., and the objects being detected does affect the accuracy numbers. Certain models just seem to do better with certain objects. The Mask R-CNN model seems to do very well with people being able to detect a person accurately if even the smallest part of any part of the person was in the image.

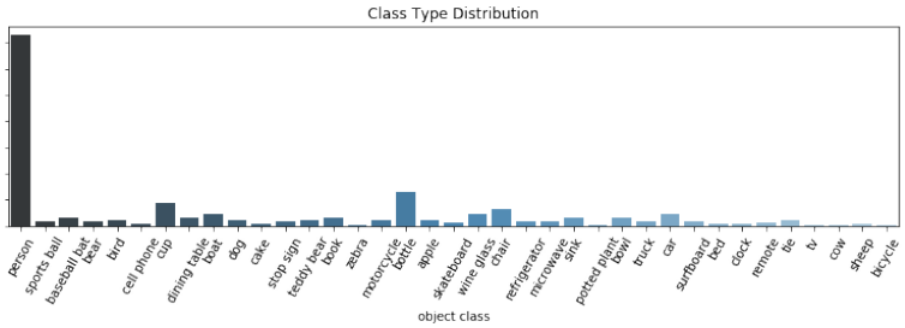

Fig. 7. Object Class Types

5.1 Grid Annotation

The ability of the algorithms to locate the objects on the grid is absolute in that if the object touches an area, then that area is output as the annotation. It is expected for this to be 100 percent accurate and this is the case.

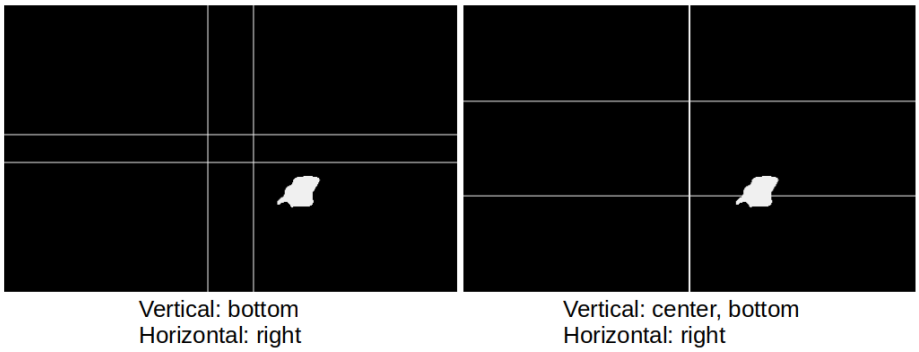

Fig. 8. Sample Location Output

Though both types of grids have similarities, there are differences between them. The grid with the vertical and horizontal output is always 3 X 3 and specifies the percent size of the middle regions. The other grid system allows the specification of as many squares as needed and tags each square with an absolute position. The former grid does its detection by looking at which regions the actual pixels of an object are touching. The later grid uses the bounding box to determine which grid coordinates are being touched. This means with oddly shaped objects, blocks can be flagged that do not actually have any pixels for the object. Figure 8 and 9 show sample outputs for these two systems.

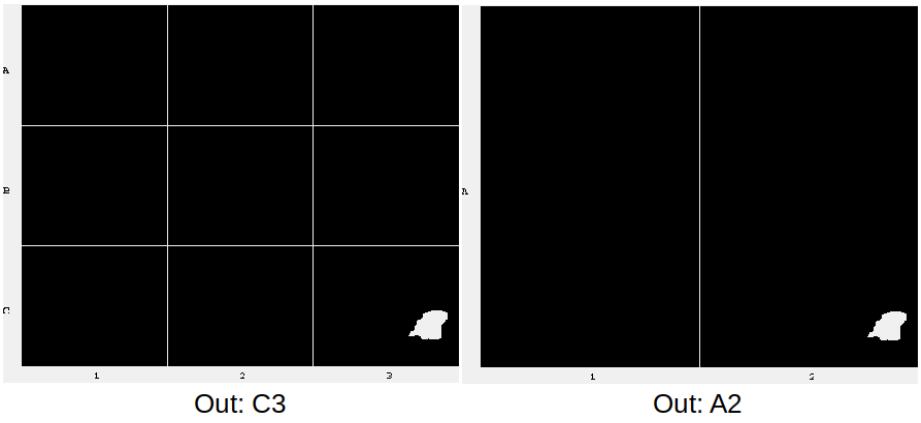

Fig. 9. Sample Grid Output

5.2 Object Relations

Because of the issues with depth, the accuracy was measured both taking depth into account and ignoring depth altogether. The results can be seen in Figure 10. These results are deceiving and not quite as good as they look. This is because there is an inordinate amount of negative object detections versus positive. The program will evaluate six different types of relationships, and if it only finds one or none, the unused detections are usually evaluated as True Negatives.

For example, if an image only has a man and a dog standing next to each other not touching with nothing else in the image, then five relations automatically become True Negatives which artificially inflates the accuracy score. The actual breakdown between positive and negative detections for each relation type can be seen in Figure 11. In order to get a better estimation of the performance of the program, precision is used in order to only evaluate the positive detections. This result can be seen in Figure 12. Though it looks like 'in' and 'on' have better accuracies, they are detected much less than the other relations. Even 'next to' which has the largest share of positive detections is still barely more than a quarter of all possible detection evaluations if accuracy is being evaluated.

When the precision numbers are evaluated, much bigger differences can be seen between ignoring depth and taking depth into account. In particular, 'below', 'above', and 'next to' without depth perform respectably, but all three with depth are below 50% precision. With no depth, 'touching' predicted contact reliably on the test data. Since this data set features more people than vehicles or animals, there was not as much of an opportunity to evaluate 'on' or 'in'. Though it looks like the performance is comparable, with the amount of detections these had, it is not expected that this performance would be the same if the images featured peoples in or on cars more prominently.

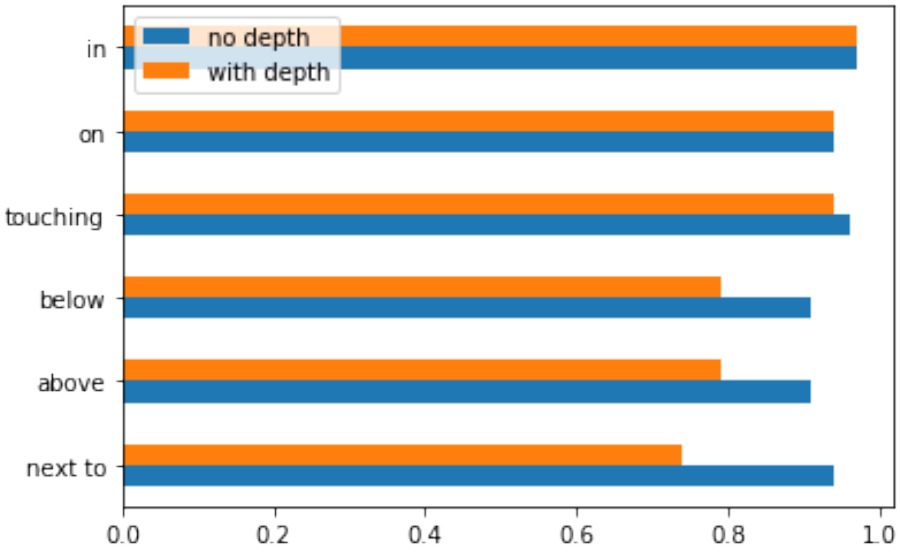

Fig. 10. Relationship Accuracy

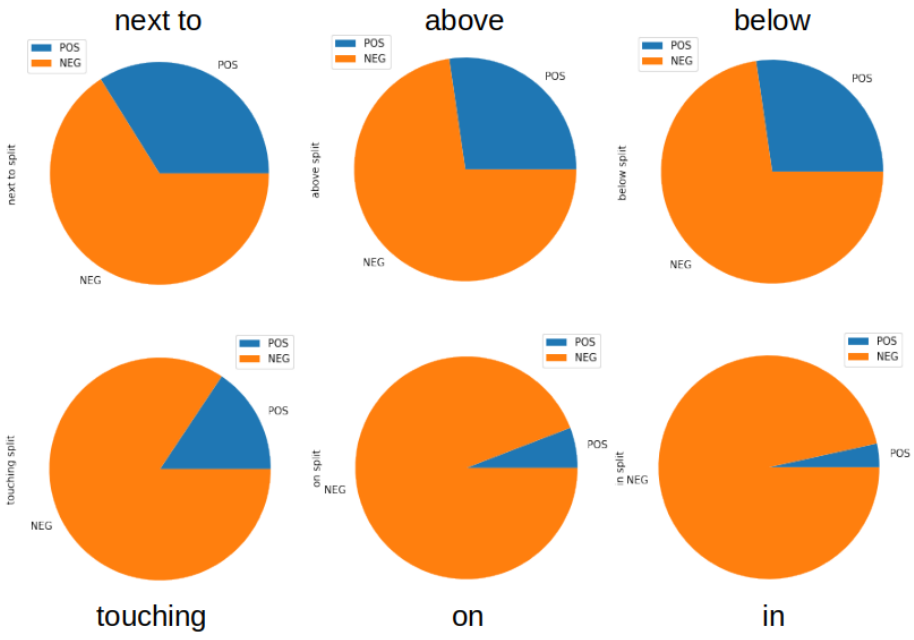

Fig. 11. Relationship Positive/Negative Splits

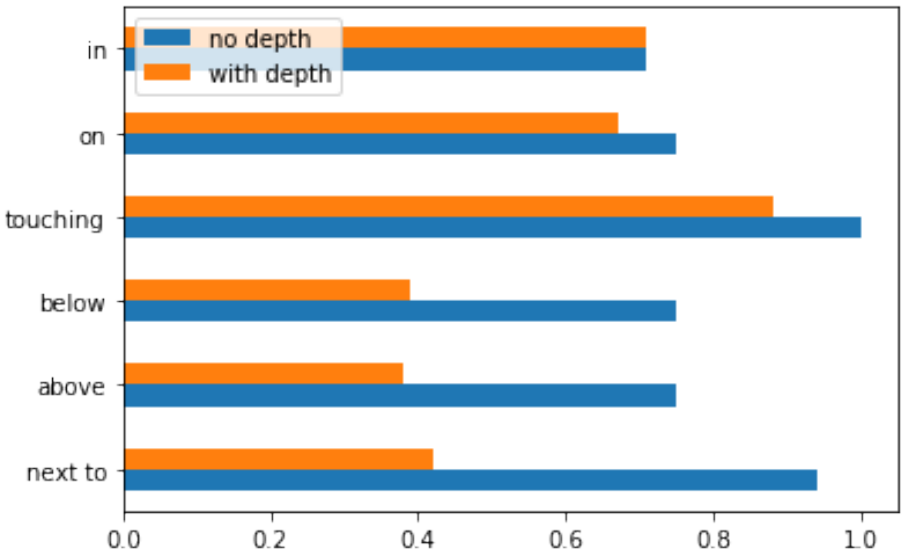

Fig. 12. Relationship Precision

6 Ethics

Though, an algorithm that can generate relationships of objects in an image can be a useful tool, like any tool, it can also be a weapon. Algorithms empower artificial intelligence. They are the foundation behind the decision making process' accuracy and efficiency. However, this only means that algorithms ultimately give humans a more accurate and more efficient way of implementing bias. Being able to better discern images based on the relationships of objects not only allows more specific image searches, but it in general provides a better way to discern or identify or label any visual that can be digitized. Performing this task to filter images is powerful, but if applied incorrectly can be discriminatory. With images, there may be an acceptable tolerance for any errors, but it is unethical to consider the cost when the rights or even well-being of an individual are at stake. The trust that is placed in artificial intelligence and the algorithms behind them creates a greater potential to begin this line of thinking. Another important issue has to do with the fact that these algorithms can inspire an unwarranted trust in their identification ability. The models, though good, are not perfect, and they can still struggle if there is for example insufficient lighting. This could lead to a human making a judgement call on the computer identification. If there is not any transparency in a given system, the human judgement call would never come into question.

7 Conclusions

Much information can be determined from the pixel and bounding box outputs of the Mask R-CNN. They can be used to determine the absolute location of a detected object in the image. To a lesser degree, they can also be used to determine the relationship of a detected object to other detected objects. Part of the issue with relative relationship determination using only bounding boxes and masks is the variability created by all the different types of camera angles, plus the variability in the different sizes of objects being detected, as well as the fact that the detection is good enough to recognize only partial pieces of an images as a whole. Several pictures had just a hand or foot recognized as a person. Being able to reliably describe a relationship between objects would require collection of many different types of images of that particular object. This could aid in helping a rule based method like this. However, the absolute relationships can reliably be associated with keywords which can be used to more precisely filter large banks of images. The relative relationships tags can be used as well, but dealing with the issues of depth, makes this model perform at best in the 50% precision range. This reduced performance, however, may be sufficient enough to help in sorting through large banks of images.

References

1. Bowen, C.e.a.: Revisiting rcnn: On awakening the classification power of faster rcnn. In: arXiv.org (2018)
2. Burkov, A.: The Hundred-Page Machine Learning Book. Andriy Burkov (2019)
3. Gao, J.e.a.: Note-rcnn: Noise tolerant ensemble rcnn for semi-supervised object detection. In: arXiv.org (2018)
4. Johnson, J.: Adapting mask-rcnn for automatic nucleus segmentation. In: arXiv.org (2018)
5. Lei, Q., e.a.: Research on human target recognition algorithm of home service robot based on fast-rcnn. 10th International Conference on Intelligent Computation Technology and Automation (ICICTA) **2017**, 47–55 (2017)
6. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context (2014)
7. Liu, Y.e.a.: A method for singular points detection based on faster-rcnn. In: Applied Sciences 8.10 (2018) (2018)
8. Malhotra, K.R.e.a.: Autonomous detection of disruptions in the intensive care unit using deep mask rcnn. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops **2018**, 1944–1946 (2018)
9. Sommer, L.e.a.: A-fast-rcnn: Hard positive generation via adversary for object detection. In: 018 25th IEEE International Conference on Image Processing (ICIP). p. 3054–3058 (2018)
10. Sorokin, A.: Lesion analysis and diagnosis with mask-rcnn. In: arXiv.org (2018)
11. Sun, Xudong, W.P., Hoi, S.C.: Face detection using deep learning: An improved faster rcnn approach. In: Neurocomputing 299.C (2018). p. 42–50 (2018)

12. Ullah, A.e.a.: Pedestrian detection in infrared images using fast rcnn. In: 2018 Eighth International Conference on Image Processing Theory, Tools and Applications (IPTA). pp. 1–6 (2018)
13. Wang, Xiaolong, S.A., Gupta, A.: A-fast-rcnn: Hard positive generation via adversary for object detection. In: arXiv.org (2017)
14. Yu, Y.e.a.: Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn. Computers and Electronics in Agriculture **163** (2019)
15. Zhang, W.: Characters detection on namecard with faster rcnn. In: arXiv.org (2018)
16. Zhang, Changzheng, X.X., Tu, D.: Face detection using improved faster rcnn. In: arXiv.org (2018)