iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** I **Month of publication:** January 2022

DOI: https://doi.org/10.22214/ijraset.2022.40135

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue I Jan 2022- Available at www.ijraset.com

Amenity Detection using Machine Learning Algorithm

S. Sivaranjani[1], KG. Haarish Kannan[2]
[1]*Panimalar Institute of Technology*
[2]*Jeppiaar Institute of technology*

Abstract: ***For many centuries, statistics have been used as a prediction tool, until recently data science evolved. When hunting for a house, you may want to check out all the amenities the house has, to know if the house is really worth the price mentioned in any real-estate dealer website. To find so, image classification models are widely in use which first categorizes the photos listed on the website into different rooms and then finds the amenities present in the room by object detection algorithm [2]. Through Computer vision, we get a better understanding of the image/video which eases our house hunt. A comparative study between machine learning algorithms used in the state of art model and my proposed model for detecting amenities with a higher precision rate is produced in the upcoming sessions.***

Keywords: Dataset, Machine Learning

I. INTRODUCTION

What is an amenity? A sofa in the living area, a shower in a bathroom, in simpler terms, Any useful object in a room is an amenity. When listing our house for rent on any broker website, we may be too careless to mention some of the all amenities present in the house. That is where object detection comes in as a helping hand.

Object detection is the primary key area of computer vision and has evolved over the years. From detecting a cat from a picture to detecting the covid-19 protocol compliance, computer vision has unfolded the map to hike up the profitable business ventures [2],

A. Object Detection in Finding Amenities in a Room

When you upload an image on any real-estate website, a computer vision machine learning model scans the images, finds the amenities and will add the amenity to the list automatically [1][3]. An example of how an object detection algorithm works in finding the amenities in a room is shown below.



This raw unfiltered image contains numerous amenities like refrigerator, table, dishwasher, oven etc, which may go unnoticed by the seller or the buyer.



In the above image, the object detection algorithm detects all the amenities and draws a rectangular wire frame which is to be listed in the website for your occupant to look upon. Even if you have missed to add anything up in your list of amenities, machine learning will be your rescuer thereby increasing the sale value of your house [5].

©IJRASET: All Rights are Reserved

B. Supervised Machine Learning

Supervised machine learning allows us to gather data and produce output as a result of past experiences thereby enabling us to solve various types of real world entities [3]. It enables us to make the required changes to the algorithm for it to learn correctly. One such is the amenity detection for estimating the real-estate prices [1]. Due to the limitless amount of data set present online, it'll be more feasible to use supervised learning over any other methods.

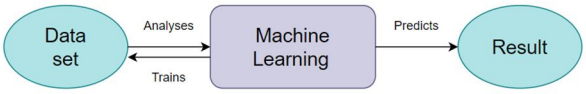
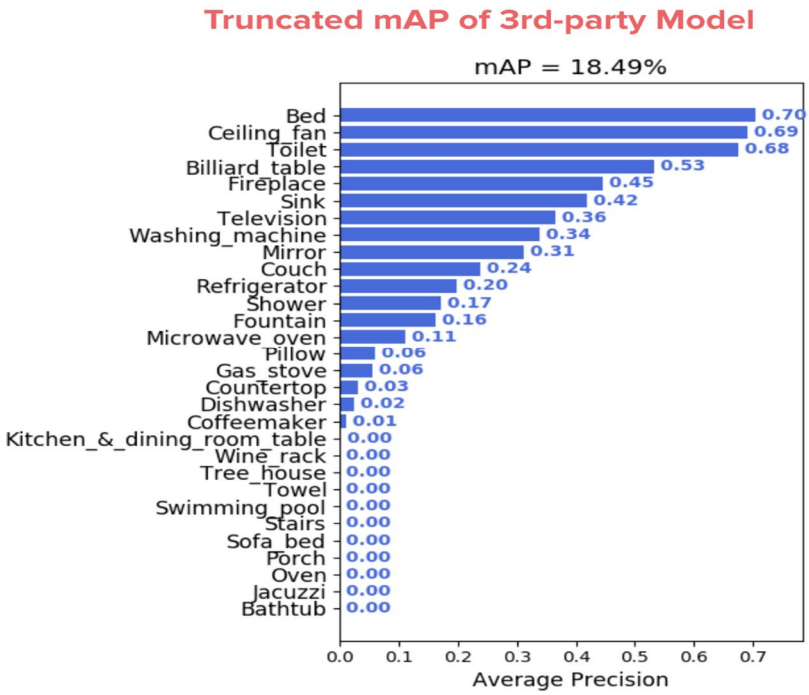


Fig.1 Process of Machine learning

II. SYSTEM ANALYSIS

A. Existing System

Existing system fails to have the maximum precision in detecting the amenities.



As you can see, some key amenities like bathtub, jacuzzi have very low or zero precision values. Even the best performing classes have low precision value which might cause the sale price of the real-estate to drop drastically.

B. Proposed System

The ultimatum of the proposed system is to not only increase the worst performing classes like dishwashers but to also increase the precision rate of the best performing classes like fireplace. Steps involved in increasing the precision:

1) Identify the classes in the input image.
2) Preparing data
3) Evaluating algorithms
4) Improving Results

Increasing the precision value is achieved by implementing the following machine learning techniques:

- SSD MobileNet - V2
- Faster RCNN ResNet V2

III. SCOPE

Nowadays, a great number of decent solutions have already emerged, some of which require minimal efforts, like this one trying to understand the risk factor behind low precision values. The scope of this paper is to increase the exactness of amenity detection in object detection algorithms using machine learning techniques.

A. Objectives

1) Data validation.
2) Data Preparation.
3) DataVisualization.
4) Using algorithms for comparing the accuracy.

B. Data set

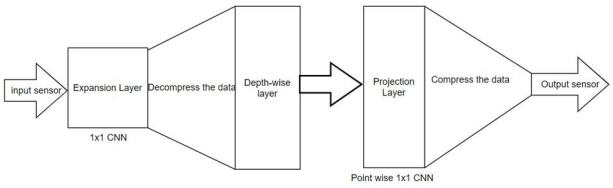
Supervised learning method is used due to the huge availability of open-source data. The latest version of open image dataset v6 is used since it contains approximately 9 million annotated images. It has a total of 600 image classes making it one of the largest dataset available online. The images are filtered based on object classes that are relevant to the real estate properties and amenities in a room [4]. After heavy filtration, there were about 40 classes from which the dataset was built. The dataset consisted of about 60k images by itself and each class consisted of about 1.6k images on an average.

C. Methodology Used

Object detection models that are trained using Machine Learning algorithms typically require massive amounts of data [3]. The standard is to have a few thousand images per feature class. The dataset which we are going to use to train our model on has about 60k images in total and 40 feature classes, each containing about 1.6k images. This is more than enough to train our model on. The model will be trained through the '*Supervised Learning*' approach. Two pre-trained models are used on top of that for optimizing it. They are :-

- SSD-MobileNet V2
- Faster RCNN + Inception ResNet V2

1) SSD-MobileNet V2: SSD-MobileNet V2 is a single-stage object detection model which has been trained from the Common Objects in Context (COCO) dataset. This model has three convolutional neural network layers. The first and last layers are 1x1 convolutional whereas the second layer is a depthwise convolutional layer. The purpose of the first layer is to expand the number of channels in the data before it gets sent to the inner layers. For this reason the first convolutional layer has more output channels than input. So this leads to the input and output of the block to be of low-dimensional tensors whereas the processing that happens inside the block will be done on high-dimensional tensors.We can set the amount of data that needs to be expanded through one of the hyperparameters. The second depthwise layer filters the input from the first. The third neural network which is a point wise 1x1 convolution is responsible for making the final output channels smaller. It does this by passing a high-dimension data into a tensor with a low number of dimensions.

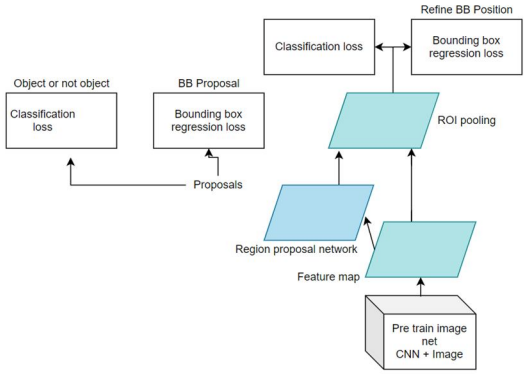


Similar to ResNet, SSD-MobileNet V2 also has Residual Connections. They help with the flow of gradients throughout the network. It should be noted that the residual connections will only be used when the number of input channels is equal to that of the output channels.

Each layer in SSD-MobileNet V2 has ReLU6 activation function and batch normalization function. The third convolutional layer does not have any activation function associated with it. The architecture consists of a total of 17 building blocks in a row which is followed by a 1x1 convolution, global average pooling layer along with a convolution layer.

This is why SSD-MobileNet V2 is one of the most efficient and powerful object detection models.

2) *Faster RCNN + Inception ResNet V2:* Faster-RCNN neural network works very similarly to RCNN. But due to some key changes we get better detection results. Faster-RCNN does feature extraction over the image before finding regions of interest. This makes the model run only one CNN per image rather than running 1000 CNN's over 1000 regions of overlaps [4]. Faster-RCNN also replaced SVM with a Softmax layer, this helps to extend the neural network's prediction instead of having to create a new one.

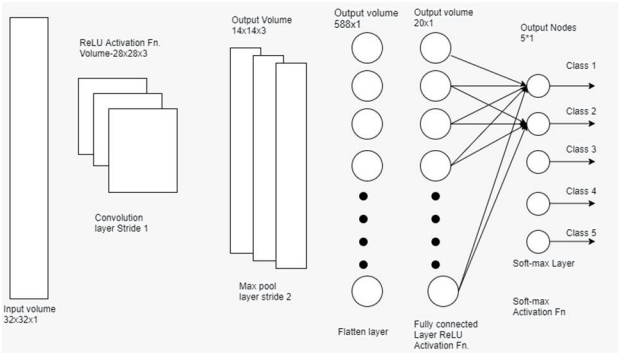


The Region Proposal Network (RPN) which is the neural network which is responsible to propose regions from an image is a key role player in Faster-RCNN's performance. The last layer of RPN has a 3x3 sliding window which moves across the feature map and transforms it into a lower dimension. For each location of the sliding window, it generates several possible locations. The amount of locations can be set through a hyperparameter. Each proposed region has an "objectness" score and coordinates representing the bounding box of the proposed region. After obtaining the proposed regions from RPN, we feed it directly into a Fast-RCNN along with a pooling layer, Softmax classification layer and a bounding box regressor. Faster-RCNN is one of the best and widely used object detection models. It also boasts very high speed and state-of-the-art accuracy.

D. Convolutional Neural Network (CNN)

In mending a vacuum between human skills and machinery, Artificial Intelligence has seen a tremendous rise. In order to make wonderful events happen, researchers and fans are working on several areas of the subject. The Computer Vision realm is one of several such fields. The target in this area will allow the computer to observe the world like people, to comprehend and even to utilise knowledge for many activities, such as image & video recognition, image analysing & classification, media recreation, referrals, natural language processing, etc [4]. Computer Vision advancement with Deep Learning has been built and improved through time, principally through one specific method which is the Convolutional Neural Network.

In comparison with other different classifiers, the preprocessing involved in a Convolutional Neural Networks is considerably less. While hand-made filters with just enough training are manufactured in the rudimentary techniques, ConvNets is able to learn these features.



The Convolutional Neural Network's design is similar to that of the connection network of neurons in the human mind and was influenced by the visual cortex arrangement. Individual neurons react to stimuli only in a small visual field area known as the receptive field. The whole visual region is covered by a combination of such fields.

Various CNN designs are available, which have been fundamental to the construction of algorithms that would in future enhance AI in their entirety. A glimpse of some very important architectures are:

1) LeNet
2) AlexNet
3) VGGNet
4) GoogLeNet
5) ResNet
6) ZFNet

IV. WORKING

Computer vision is the field under *Artificial Intelligence (AI)* which deals with making the computers understand real life images and videos rather than just looking at them as a collection of colors for every pixel. Image classification is a sub-field of *Computer Vision (CV)* which focuses on labelling and categorizing groups of vectors or pixels within an image or a frame (in case of a video) based on a predefined set of specific rules. This can be achieved by two different ways, *supervised* and *unsupervised*. Object Detection falls under Image classification and it deals with identifying objects of a specific set of classes in images and videos.

This paper aims to come up with a novel Object Detection algorithm which detects Amenities present in an image and outperform the existing state-of-art Amenity Detection model available right now.

A. State-of-the-art Model

Object Detection technology has evolved rapidly compared to its initial days. There are a number of 3rd party API's which are used widely to build a generic object detection model. They are also quite effective and cheap to deploy. Integrating these API's in pre-existing products is also hassle-free.

We began testing the current 3rd party state-of-art models and noticed some flaws. The models identified amenities well and good, but the predicted labels were far too vague. For example, in one instance from the sample space, an 'oven' was identified and labelled as 'kitchenware'. Though this is technically true, in some applications especially from the real estate sector it would prove more useful if the object got labelled with its specific name rather than a generic class. We identified that one of our primary objectives will be to increase the amount of labels and make the model be able to detect amenities in a much more scrutinized manner. After testing out 3rd party API's. I moved on to test pre-trained models from open-source projects such as Detectron Model Zoo and Tensorflow Detection Model Zoo. The above flaws were seen once again here, where the models were not capable of labelling the amenities detected properly. Along with that certain labels which were predicted were a bit far off and the overall Precision score was lower as well.

From the above inferences, it was clear that a customized model should be built to improve the amenity detection quality as pre-trained models were not upto the mark [1].

The elephant in the pond right now was the poor labelling quality and without proper labelling instructions building a customized model will not be as fruitful. The first step will be to build a customized set of amenity labels and structure the image dataset around it. After annotating the images with the updated labels, the model can be trained against it. This should improve its performance and make it capable of labelling the identified instances in a more specific manner.

B. Describing the Taxonomy

Taxonomy is the process of naming, describing, and classifying the much needed labels for annotating the dataset. The taxonomy should be described in such a manner that it self-explains our interest and the primary objective of the model. Since, in this case, the sheer amount of labels that are needed are so vast, not to forget that it also grows day by day, meeting the pre-said rule-of-thumb is highly questionable. Instead of building the taxonomy of every possible amenity available, a minimal viable list of the utmost basics was crafted. The idea was to expand upon the taxonomy when the model itself proves to be fruitful. Choosing this minimal list proved tougher than initially thought, since the importance of an amenity did not match with the amount of images that was available in the dataset which later could be annotated with it. So the goal was to come up with an initial taxonomy which was as minimal as possible which will lead to a dataset as big as possible. After coming up with one, the strategy was to keep iterating the taxonomy fast until a good precision score is achieved.

The Open Image Dataset V6 has around 9 million annotated images with object bounding boxes (OBB), image-level labels with visual relationship representation.

©IJRASET: All Rights are Reserved

The OBB labels available in the dataset had a whopping 600 object classes, obviously we are not going to use them all in fact we are only interested in a very minor fraction of it. These labels also followed an hierarchical structure, on the top-most level the following labels were available - Clothing, Food, Equipment, Tool, Vehicle, and along with them were Appliances, Furniture, Electrical Appliances, etc… The next step was to identify OBB classes which are related to amenities so that images from those classes can be collected to build the dataset. After manually reviewing all the classes available in the dataset (~600), a total of 40 classes were picked which proved useful and relevant to the desired dataset. Some of the labels that were selected were oven, refrigerator, jacuzzi, tree house, swimming pool, bed, couch, etc..

E. Constructing the Dataset

Building an image dataset from The Open Image Dataset V6 should have proven easy except it was not. Even though it consisted of around 9 million images, these images were not distributed evenly among the labels available in fact the distribution was highly imbalanced and biased. Certain classes had millions of images while others barely had any comparatively.

Unfortunately many of the classes which were curated earlier fell in the minority category of The Open Image Dataset V6. After scraping enough data, the dataset consisted of 100k class object instances over 60k images.

Just like previously said, supervised machine learning models need loads of data to learn. It is general practice to have a few thousands of samples within a single class so that decent model performance can be achieved. 60k images for 40 object classes meant, on average each class will have around 1.6k images which is sufficient.

F. Feature Engineering

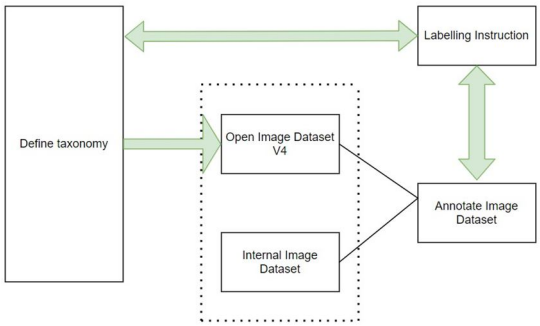
Feature engineering is a big part when it comes to building machine learning and deep learning models [5]. It not only ensures we use the available dataset to its maximum potential but also that future iterations of the dataset will learn more effectively. Feature engineering uses domain knowledge to create more features in the dataset based on the existing ones. These features will be used by the models to learn from the dataset. The more features a dataset has, the more effectively it can learn from it.

Image annotation assigns metadata to the image. These annotations will boost the model's learning efficiency by a huge factor. The metadata typically consists of information in the form of location marker, keywords, captions and much more. When we annotate the images with useful metadatas as such, we enable the model to learn from these information as well, hence improving its overall efficiency [11].

Image annotation services are provided by many third-party vendors as they are crucial for any object-detection model workflow. The end-user would have to provide the vendor with labelling instructions and the raw data, using which the vendor will label the data and return those annotations. The key element here is the labelling instruction, the more specific it is the much better annotations we get. It is near impossible to write the perfect labelling instruction at the beginning. so an iterative approach was opted [5].

The Google Data Labelling service was chosen for annotating images due to the following reasons :-

- It supports nearly 100 classes for labeling.
- Great UI/UX which eases the workflow and lets the user monitor the labelling progress.
- Questions and Feedbacks were frequently asked to improve the labelling instructions due to which niche scenarios were met without any hindrance.



As mentioned previously, to incorporate an iterative approach in building the labelling instruction, small batches of data were sent to the Google Data Labelling service instead of in bulk. This made updating the labeling instructions after each batch possible, so that the majority of the dataset had better annotations. The initial batches can be re-annotated using the updated and improved labelling instructions if needed to. Thanks to this approach, certain amenity classes in the dataset were found to be not so helping [11]. These classes were later removed from our taxonomy, changing its size from 40 to 30. With the improved taxonomy, the dataset was re-annotated for maximum efficiency.

G. Training Machine Learning Model

With the dataset ready and annotated, now it is time for model training. Initially, Tensorflow's Object Detection API was used to create tfrecords from the existing dataset and then training the model. The training progress was tracked through Tensorboard.

To fine-tune the model, pre-trained models were used. The options considered were SSD MobileNet V2 and Faster RCNN Inception V2. Both of them were stellar choices but they had their own pros and cons. To sum up, SSD MobileNet V2 was fast but had a lower accuracy whereas Faster RCNN Inception V2 was a little slow but obtained great accuracy [9][10]. In a sense, both the models produced opposite results to each other. Since both time and accuracy are important and one cannot be called superior compared to the other. A simple benchmark was set up to determine which model truly performed better than the other. A sample dataset containing about 10% of the existing dataset (8k Images with 30 classes) was created for this purpose. mean Average Precision (mAP) was used for evaluation purposes [7].

Faster RCNN Inception V2 achieved an mAP of 28% whereas SSD MobileNet V2 only managed to achieve a meager 13%. This was highly due to the fact that Faster RCNN Inception V2 has higher accuracy than SSD MobileNet V2 [11]. But on the other hand, SSD MobileNet V2 is faster than RCNN Inception V2 and will be very handy when it comes to scalable applications where accuracy wouldn't be a big deal [9][10]. Since the benchmark failed to give a declarative conclusion, it was decided to use both of the pre-trained models and let the two final versions of the Object-Detection Model decide which one to stick with.

Annotation generation from Google Data Labelling service was stopped to start generating the dataset for model preparation. The default learning rate of SSD MobileNet V2 and Faster RCNN Inception V2 were different to each other. SSD MobileNet V2 had a higher rate to that of Faster RCNN Inception V2 [6][10].

- SSD MobileNet V2 - *8 e-4*
- Faster RCNN Inception V2 - *6 e-5*

The learning rates were reduced approximately by 10 % since we are trying to implement transfer learning. If the learning rates were not reduced the gradient update might end up too large and make our end-model develop *'bad intelligence'* and render it useless. Number of steps was also reduced to 1M for the above mentioned reason. The model was trained on a Tesla K80 single-core GPU [9].

Initial inference while training the SSD MobileNet model was that the loss function deteriorated rapidly in the beginning. After training for 4 days, approximately 100K steps the decline stopped and an upward trend was seen. Due to this weird behaviour training was stopped on the 5th day. To top that off the progress of training SSD MobileNet V2 was dead slow as well. In the end, mAP was increased from 13% to 22% [10].

A similar story with Faster RCNN Inception V2, the loss function was small in the beginning but started to show erratic behaviour as time passed. One clear difference was it had a better training time compared to SSD MobileNet V2. But in the end, the mAP of the model had negligible improvement [9][10].

The goal was to have at least 50% mAP to prove better than the state-of-art. But clearly both the models, SSD MobileNet V2 and Faster RCNN Inception V2 failed to even remotely come closer to the target. Unable to come up with a plausible hypothesis for this behaviour, more time was invested in feature engineering as better data will always yield better results no matter what [6]. Iterating the above methodologies once more yielded results that were either negligibly positive or had no impact at all. After investing more time in diving under the hood of both our pre-trained models, an hypothesis was finally made. It was highly possible for the loss function to be stuck at some local minima, but the problem was that fixing this blocker will be very tedious. The sheer amount of time required for debugging is huge by itself let alone fixing it. Switching to a new model sounded both exciting and easy at this point, so these two pre-trained models were stopped here to revisit in the future after building a model that succeeds at least the basic expectation of this research work.

To proceed with the research work, Google AutoML Vision was decided to be used. Google AutoML vision is an automatic ML development cloud service which trains ML models based on the image dataset that you provide along with the labelling if possible.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue I Jan 2022- Available at www.ijraset.com

The results were much better than what was anticipated, 80k images were provided to the service and within the next few days a high-accuracy model was developed which passed the basic expectations of this research model.
The plan right now was to polish up the rest of the other areas of our architecture with the AutoML model as a base and come back to build a better 3rd party ML object-detection model later.

V. MODEL EVALUATION

A. *General Definitions*

1) *True Positive (TP)* - True positive is the outcome in which our AI model correctly predicts the positive class as outcome.
2) *True Negative (TN)* - True negative is the outcome in which our AI model correctly predicts the negative class as outcome.
3) *False Positive (FP)* - False Positive is the outcome in which our AI model incorrectly predicts the positive class as outcome.
4) False Negative (FN) - False Negative is the outcome in which our AI model incorrectly predicts the negative class as outcome.
5) *Accuracy formulae* - (TP + TN) / (TP + FN + FP + TN)

True Class	Positive	Negative	Total
Positive	TP	FN	TP+FN
Negative	FP	TN	FP+TN
Total	TP+FP	FN+TN	TP+FP+FN+TN

6) *Precision* - Precision is the measurement of correct outcomes out of all the obtained outcomes.

$$\text{Precision} = TP / (TP + FP)$$

7) *Recall* - Recall is the measurement of the proportion of correct outcomes which were actually obtained.

$$\text{Recall} = TP / (TP + FN)$$

8) *F1 Score* - The F1 score combines the AI model's precision and recall to give the true accuracy of the model.

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

B. *Model Evaluation*

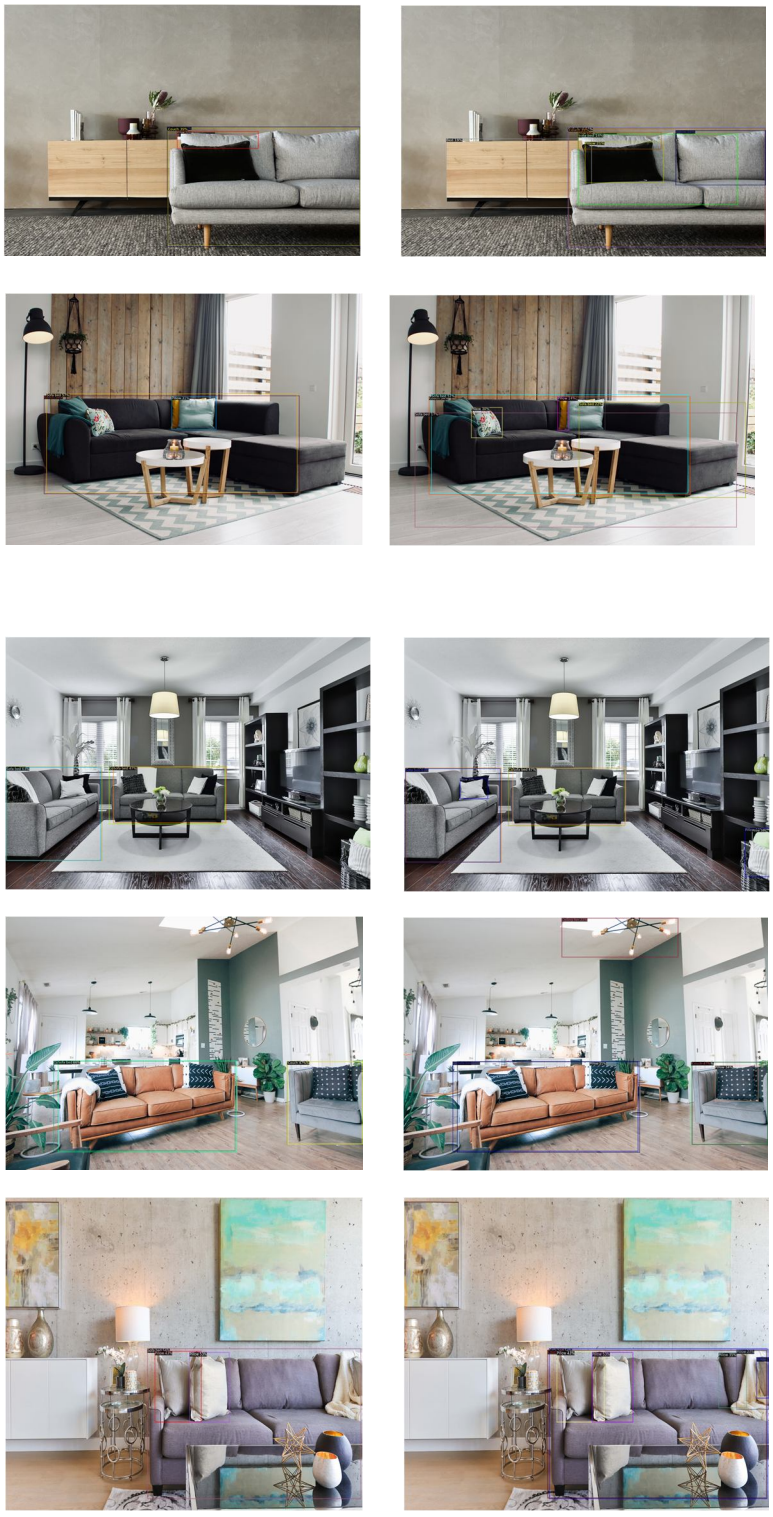
Evaluating the built machine learning model is very essential as that will not only give insights on how our model is performing but also identify potential areas of improvement [5]. The Google AutoML model already achieved the least expected mAP but it is equally more important to evaluate other aspects of the model.
A local evaluation of the model with 10% of the held-out data (about 7.5K images) achieved an mAP of 70% which was much higher than anticipated result and the least criteria to be met. It is important to note that the above said mAP was average out of all the available labels. Certain labels which were also highly domain-important labels such as *lavatory, cot, bed and swimming_pool* achieved above 90% mAP. Some of the worst performing classes were *tree_house, dishwasher, jacuzzi and wine_rack* but even these labels managed to achieve a mAP of 45%. But these classes cannot be left as such because such amenities improve a real-estate's value by a huge amount and should have acceptable accuracy. The mAP achieved for the classes was directly proportional to the amount of data present in the training dataset. So more data sets of the labels were added and the model was re-trained to improve its mAP for those particular classes [6].
Another important inference which was observed during the offline-local model evaluation was that the mAP value was very sensitive to the test-train split ratio, that is its value varied a lot even to a small disruption of change to the test-train split ratio. And the instability of mAP was purely due to the poor performing classes and not because of all classes as a whole. For example the median of Average Precision of all classes was 83%. Again this issue was fixed after re-training the model with enough data sets for the low performing classes [7].

C. *Model Performance in Real Time*

Here is the model in action, sufficient examples are given where both the current state-of-art model and our customized generic 3rd party model are seen in action. One important conjecture that needs to be announced is that this research work model only has 30 classes whereas the state-of-art model has many classes. This was done to reduce the amount of training time involved and not because it was not possible. The amount of classes involved with the custom model can very well be scaled up if necessary data sets are available.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue I Jan 2022- Available at www.ijraset.com



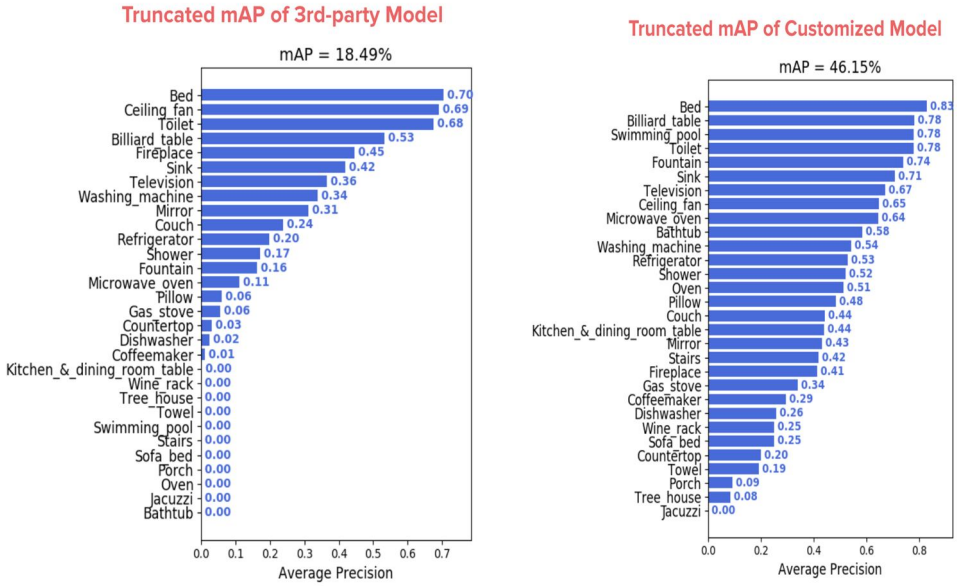
Clearly, the custom generic model performs leagues better than the state-of-art model. Even though it only covers 30 classes, it provides an exponentially higher accuracy than the state-of-art model.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue I Jan 2022- Available at www.ijraset.com

D. Differences Between State-of-art and Research Work

To understand why the research work model does better than the state-of-the-art, it's important to understand their differences. The taxonomy defined for the research model is not similar to that of the state-of-art model, as the taxonomy pretty much sculpts the dataset which will be used to build the model. This is a very important point to be considered while talking about the differences between the two models. The state-of-art model only displays predictions with a confidence score higher than 0.5, which completely negates the right-side of the model's precision-recall curve where the recall is really good but the threshold value is lower. This lowers the mAP of the state-of-art model, though not by that much but still the end mAP we see of the state-of-art model is not the exact one. But to truly prove that the research model performs better than the state-of-art model, the right part of the research model's precision-recall curve was truncated in the same manner and the mAP was calculated [7]. The research model scored a mAP value of 46% and the state-of-art model scored 16%, which proves yet again that the research model performs better and more efficiently than the state-or-art model. And finally, just like previously touched, the dataset which is used to train the model determines the performance of a model by a huge factor. Important domain-specific datas were included in the research dataset and clever feature engineering was implemented to get even the slightest advantage possible out of the model. This made the model more used and adaptive to its nature of input and hence improved its efficiency and performance.



VI. CONCLUSION AND FUTURE SCOPE

Amenity detection is a subset of Object-Detection just like previously said. It would be nice if we were able to transfer learning from a generic object-detection algorithm to create low-end models that are focussed on a specific objective. Plus by using a generic object-detection model over an amenity detection model we will be able to obtain information that are important for other secondary features in the pipeline. For example, noticing a beach from a photo intended for a bathtub will prove useful to the value of a real-estate [2]. Dealing with data cleaning and image noise control covers the major part of ML and Data science works. A job that is considered tedious. But luckily, technology has grown so much today that it is possible to automate such tasks at least to an extent by tapping into the power of AI. Images can automatically be post-processed to lower their size and make them more ML friendly. This becomes highly needed when the database grows to a whooping size so that money cannot be wasted in buying more storage or time lost while training the model. Another area of interest will be image quality control, which will help you reduce the amount of noise that is present in your dataset. A simple technique is to have two metrics for each image - Aesthetic score and Technical score. Aesthetic score covers the overall appeal of the image while the Technical score covers the variety of specificational details in the image. Thus when an image has higher Technical score but lower Aesthetic score, it can be ruled out as noise without any doubt. Like this one below

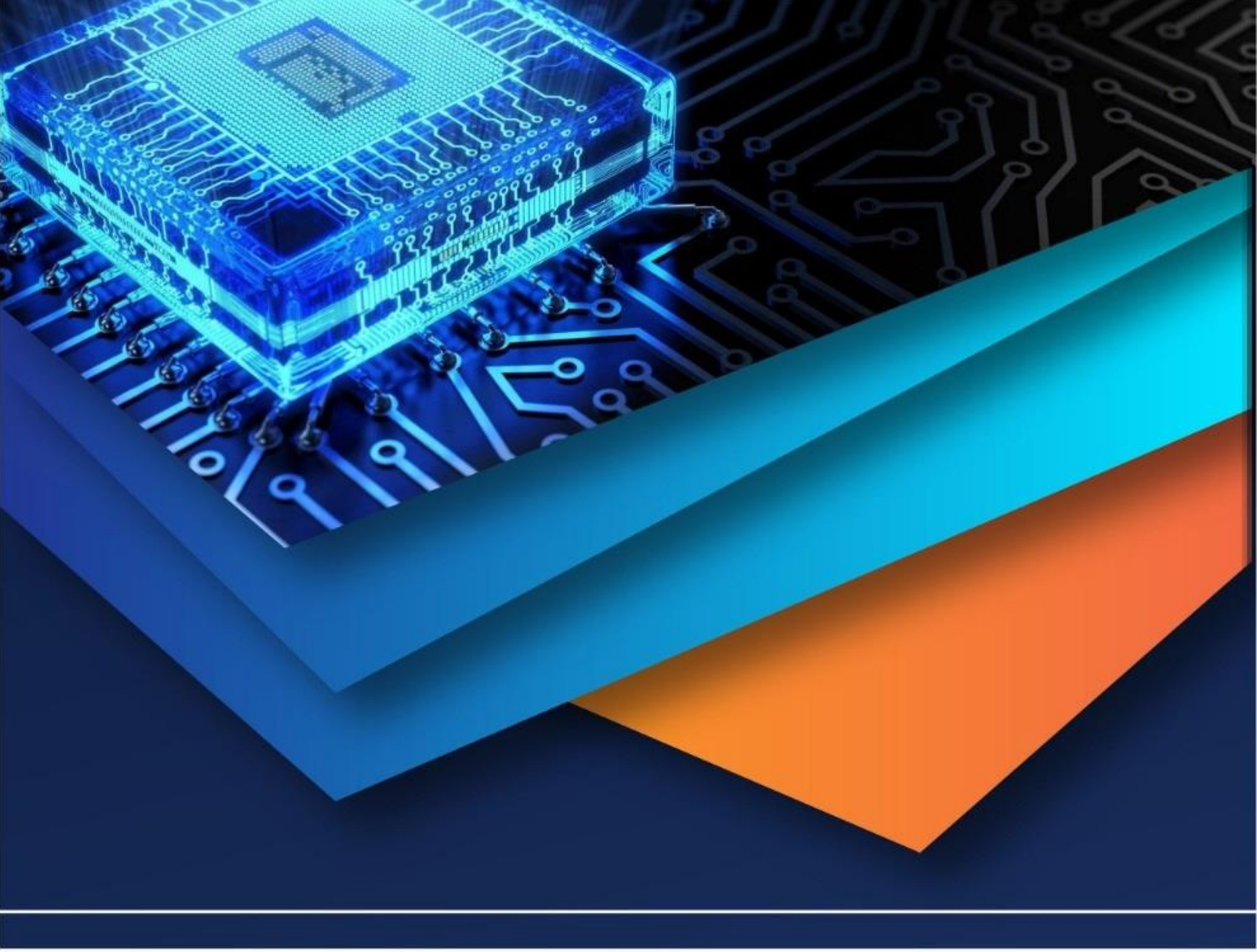
©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



The world has started to use computer vision in almost every aspect possible. This increases the need to have a good quality of specifically defined models readily available. Generic models are quite heavy and cannot be used smoothly on our day-to-day gadgets. But on the other hand, specifically defined object-detection models aren't perfect yet. This makes the future scope of this research work quite limitless.

REFERENCES

[1] Kawano, S., Imanishi, T., Ikeda, Y., Nishi, H. and Uchiyama, E., 2016. Implementation of Household's Amenity Maintaining System Based on Behavior Estimation. Procedia Environmental Sciences, 34, pp.582-593.
[2] Solomon Cheung, Y.S. and Zhang, F., 2019, December. An Intelligent Internet-of-Things (IoT) System to Detect and Predict Amenity Usage. In CS & IT Conference Proceedings (Vol. 9, No. 17). CS & IT Conference Proceedings.
[3] Wen, Richard, and Claus Rinner. "Outlier Detection in OpenStreetMap Data using the Random Forest Algorithm." International Conference on GIScience Short Paper Proceedings. Vol. 1. No. 1. 2016.
[4] Lézoray, Olivier, Christophe Charrier, Hubert Cardot, and Sébastien Lefèvre. "Machine learning in image processing." (2008): 1-2.
[5] Mahesh, Batta. "Machine Learning Algorithms-A Review." International Journal of Science and Research (IJSR).[Internet] 9 (2020): 381-386.
[6] Arman F, Hsu A, Chiu MY. Image processing on compressed data for large video databases. InProceedings of the first ACM international conference on Multimedia 1993 Sep 1 (pp. 267-272).
[7] Wan, X., Ren, F., & Yong, D. (2019, December). Using Inception-Resnet v2 for face-based age recognition in scenic spots. In 2019 IEEE 6th International Conference on Cloud Computing and Intelligence Systems (CCIS) (pp. 159-163). IEEE.
[8] Bhatia, Yajurv, Aman Bajpayee, Deepanshu Raghuvanshi, and Himanshu Mittal. "Image Captioning using Google's Inception-resnet-v2 and Recurrent Neural Network." In 2019 Twelfth International Conference on Contemporary Computing (IC3), pp. 1-6. IEEE, 2019.
[9] Latha, H.N., Rudresh, S., Sampreeth, D., Otageri, S.M. and Hedge, S.S., 2018, December. Image understanding: Semantic Segmentation of Graphics and Text using Faster-RCNN. In 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS) (pp. 1-6). IEEE.
[10] Kumar, K. S., Subramani, G., Thangavel, S. K., & Parameswaran, L. (2021). A mobile-based framework for detecting objects using ssd-mobilenet in an indoor environment. In Intelligence in Big Data Technologies—Beyond the Hype (pp. 65-76). Springer, Singapore.
[11] Younis, A., Shixin, L., Jn, S., & Hai, Z. (2020, January). Real-time object detection using pre-trained deep learning models MobileNet-SSD. In Proceedings of 2020 the 6th International Conference on Computing and Data Engineering (pp. 44-48).



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)