35th IEEE International Conference on Software Maintenance and Evolution (ICSME2019), Cleveland, Ohio, USA, September 30-October 4, 2019 (c) IEEE 2019

Losing Confidence in Quality: Unspoken Evolution of Computer Vision Services

Alex Cummaudo*, Rajesh Vasa*, John Grundy†, Mohamed Abdelrazek‡ and Andrew Cain‡
* Applied Artificial Intelligence Institute, Deakin University, Geelong, Australia
† Faculty of Information Technology, Monash University, Clayton, Australia
‡ School of Information Technology, Deakin University, Geelong, Australia
{ *ca, rajesh.vasa, mohamed.abdelrazek, andrew.cain* }*@deakin.edu.au, john.grundy@monash.edu*

***Abstract*—Recent advances in artificial intelligence (AI) and machine learning (ML), such as computer vision, are now available as intelligent services and their accessibility and simplicity is compelling. Multiple vendors now offer this technology as cloud services and developers want to leverage these advances to provide value to end-users. However, there is no firm investigation into the maintenance and evolution risks arising from use of these intelligent services; in particular, their behavioural consistency and transparency of their functionality. We evaluated the responses of three different intelligent services (specifically computer vision) over 11 months using 3 different data sets, verifying responses against the respective documentation and assessing evolution risk. We found that there are: (1) inconsistencies in how these services behave; (2) evolution risk in the responses; and (3) a lack of clear communication that documents these risks and inconsistencies. We propose a set of recommendations to both developers and intelligent service providers to inform risk and assist maintainability.**

***Index Terms*—machine learning, intelligent service, computer vision, quality assurance, evolution risk, documentation**

I. Introduction

The availability of intelligent services has made AI tooling accessible to software developers and promises a lower entry barrier for their utilisation. Consider state-of-the-art computer vision analysers, which require either manually training a deep-learning classifier, or selecting a pre-trained model and deploying these into an appropriate infrastructure. Either are laborious in time, and require non-trivial expertise along with a large data set when training or customisation is needed. In contrast, intelligent services providing computer vision (e.g., [41–54]) abstract these complexities behind a web API call. This removes the need to understand the complexities required of ML, and requires little more than the knowledge on how to use RESTful endpoints. The ubiquity of these services is exemplified through their rapid uptake in applications such as aiding the vision-impaired [1, 2].

While intelligent services have seen quick adoption in industry, there has been little work that has considered the software quality perspective of the risks and impacts posed by using such services. In relation to this, there are three main challenges: (1) incorporating stochastic algorithms into software that has traditionally been deterministic; (2) the general lack of transparency associated with the ML models; and (3) communicating to application developers.

ML typically involves use of statistical techniques that yield components with a non-deterministic external behaviour; that is, for the same given input, different outcomes may result. However, developers, in general, are used to libraries and small components behaving predictably, while systems that rely on ML techniques work on confidence intervals[1] and probabilities. For example, the developer's mindset suggests that an image of a border collie—if sent to three intelligent computer vision services—would return the label 'dog' consistently with time regardless of which service is used. However, one service may yield the specific dog breed, 'border collie', another service may yield a permutation of that breed, 'collie', and another may yield broader results, such as 'animal'; each with results of varying confidence values.[2] Furthermore, the third service may evolve with time, and thus learn that the 'animal' is actually a 'dog' or even a 'collie'. The outcomes are thus behaviourally inconsistent between services providing conceptually similar functionality. As a thought exercise, consider if the sub-string function were created using ML techniques—it would perform its operation with a confidence where the expected outcome and the AI inferred output match as a *probability*, rather than a deterministic (constant) outcome. How would this affect the developers' approach to using such a function? Would they actively take into consideration the non-deterministic nature of the result?

Myriad software quality models and SE practices advocate maintainability and reliability as primary characteristics; stability, testability, fault tolerance, changeability and maturity are all concerns for quality in software components [3–5] and one must factor these in with consideration to software evolution challenges [6–10]. However, the effect this non-deterministic behaviour has on quality when masked behind an intelligent service is still under-explored to date in SE literature, to our knowledge. Where software depends on intelligent services to achieve functionality, these quality characteristics may not be achieved, and developers need to be wary of the unintended side effects and inconsistency that exists when using non-deterministic components. A computer vision service may encapsulate deep-learning strategies or

[1] Varied terminology used here. Probability, confidence, accuracy and score may all be used interchangeably.

[2] Indeed, we have observed this phenomenon using a picture of a border collie sent to various computer vision services.

stochastic methods to perform image analysis, but developers are more likely to approach intelligent services with a mindset that anticipates consistency. Although the documentation does hint at this non-deterministic behaviour (i.e., the descriptions of 'confidence' in various computer vision services suggest the they are not always confident, and thus not deterministic [55–57]), the integration mechanisms offered by popular vendors do not seem to fully expose the nuances, and developers are not yet familiar with the trade-offs.

Do popular computer vision services, as they currently stand, offer consistent behaviour, and if not, how is this conveyed to developers (if it is at all)? If computer vision services are to be used in production services, do they ensure quality under rigorous Software Quality Assurance (SQA) frameworks [5]? What evolution risk [6–9] do they pose if these services change? To our knowledge, few studies have been conducted to investigate these claims. This paper assesses the consistency, evolution risk and consequent maintenance issues that may arise when developers use intelligent services. We introduce a motivating example in section II, discussing related work and our methodology in sections III and IV. We present and interpret our findings in section V. We argue with quantified evidence that these intelligent services can only be considered with a mature appreciation of risks, and we make a set of recommendations in section VI.

II. Motivating Example

Consider Rosa, a software developer, who wants to develop a social media photo-sharing mobile app that analyses her and her friends photos on Android and iOS. Rosa wants the app to categorise photos into scenes (e.g., day vs. night, outdoors vs. indoors), generate brief descriptions of each photo, and catalogue photos of her friends as well as common objects (e.g., all photos with a dog, all photos on the beach).

Rather than building a computer vision engine from scratch, Rosa thinks she can achieve this using one of the popular computer vision services (e.g., [41–54]). However, Rosa comes from a typical software engineering background with limited knowledge of the underlying deep-learning techniques and implementations as currently used in computer vision. Not unexpectedly, she internalises a mindset of how such services work and behave based on her experience of using software libraries offered by various SDKs. This mindset assumes that different cloud vendor image processing APIs more-or-less provide similar functionality, with only minor variations. For example, cloud object storage for Amazon S3 is both conceptually and behaviourally very similar to that of Google Cloud Storage or Azure Storage. Rosa assumes the computer vision services of these platforms will, therefore, likely be very similar. Similarly, consider the string libraries Rosa will use for the app. The conceptual and behavioural similarities are consistent; a string library in Java (Android) is conceptually very similar to the string library she will use in Swift (iOS), and likewise both behave similarly by providing the same results for their respective sub-string functionality. However, **unlike the cloud storage and string libraries, different computer vision services often present conceptually similar functionality but are behaviourally very different**. Intelligent service vendors also hide the depth of knowledge needed to use these effectively—for instance, the training data set and ontologies used to create these services are hidden in the documentation. Thus, Rosa isn't even exposed to this knowledge as she reads through the documentation of the providers and, thus, Rosa makes the following assumptions:

- **"I think the responses will be consistent amongst these computer vision services."** When Rosa uploads a photo of a dog, she would expect them all to respond with 'dog'. If Rosa decides to switch which service she is using, she expects the ontologies to be compatible (all computer vision services *surely* return dog for the same image) and therefore she can expect to plug-in a different service should she feel like it making only minor code modifications such as which endpoints she is relying on.
- **"I think the responses will be constant with time."** When Rosa uploads the photo of a dog for testing, she expects the response to be the same in 10 weeks time once her app is in production. Hence, in 10 weeks, the same photo of the dog should return the same label.

III. Related Work

If we were to view computer vision services through the lenses of an SQA framework, robustness, consistency, and maintainability often feature as quality attributes in myriad software quality models (e.g., [11]). Software quality is determined from two key dimensions: (1) in the evaluation of the end-product (external quality) and (2) the assurances in the development processes (internal quality) [3]. We discuss both perspectives of quality within the context of our work in this section.

A. External Quality

1) Robustness for safety-critical applications: A typical focus of recent work has been to investigate the robustness of deep-learning within computer vision technique implementation, thereby informing the effectiveness in the context of the end-product. The common method for this has been via the use of adversarial examples [12], where input images are slightly perturbed to maximise prediction error but are still interpretable to humans.

Google Cloud Vision, for instance, fails to correctly classify adversarial examples when noise is added to the original images [13]. Rosenfeld et al. [14] illustrated that inserting synthetic foreign objects to input images (e.g., a cartoon elephant) can completely alter classification output. Wang et al. [15] performed similar attacks on a transfer-learning approach of facial recognition by modifying pixels of a celebrity's face to be recognised as a completely different celebrity, all while still retaining the same human-interpretable original celebrity. Su et al. [16] used the ImageNet database to show that 41.22% of images drop in confidence when just a *single pixel* is changed in the input image; and similarly, Eykholt et al. [17] recently showed similar results that made a CNN interpret a stop road-sign (with mimicked graffiti) as a 45mph speed limit sign.

The results suggest that current state-of-the-art computer vision techniques may not be robust enough for safety critical applications as they do not handle intentional or unintentional adversarial attacks. Moreover, as such adversarial examples exist in the physical world [18, 19], "the natural world may be adversarial enough" [20] to fool AI software. Though some limitations and guidelines have been explored in this area, the perspective of *intelligent* services is yet to be considered and specific guidelines do not yet exist when using computer vision services.

2) Testing strategies in ML applications: Although much work applies ML techniques to automate testing strategies, there is only a growing emphasis that considers this in the opposite sense; that is, testing to ensure the ML product works correctly. There are few reliable test oracles that ensure if an ML has been implemented to serve its algorithm and use case purposefully; indeed, "the non-deterministic nature of many training algorithms makes testing of models even more challenging" [21]. Murphy et al. [22] proposed a SE-based testing approach on ML ranking algorithms to evaluate the 'correctness' of the implementation on a real-world data set and problem domain, whereby discrepancies were found from the formal mathematical proofs of the ML algorithm and the implementation.

Recently, Braiek and Khomh [23] conducted a comprehensive review of testing strategies in ML software, proposing several research directions and recommendations in how best to apply SE testing practices in ML programs. However, much of the area of this work specifically targets ML engineers, and not application developers. Little has been investigated on how application developers perceive and understand ML concepts, given a lack of formal training; we note that other testing strategies and frameworks proposed (e.g., [24–26]) are targeted chiefly to the ML engineer, and not the application developer.

However, Arpteg et al. [21] recently demonstrated (using real-world ML projects) the developmental challenges posed to developers, particularly those that arise when there is a lack of transparency on the models used and how to troubleshoot ML frameworks using traditional SE debugging tools. This said, there is no further investigations into challenges when using the higher, 'ML friendly' layers (e.g., intelligent services) of the 'machine learning spectrum' [27], rather than the 'lower layers' consisting of existing ML frameworks and algorithms targeted toward the ML community.

B. Internal Quality

1) Quality metrics for cloud services: Computer vision services are based on cloud computing fundamentals under a subset of the Platform as a Service (PaaS) model. There has been work in the evaluation of PaaS in terms of quality attributes [28]: these attributes are exposed using Service Level Agreements (SLA) between vendors and customers, and customers denote their demanded Quality of Service (QoS) to ensure the cloud services adhere to measurable KPI attributes.

Although, popular services, such as cloud object storage, come with strong QoS agreement, to date intelligent services do not come with deep assurances around their performance and responses, but do offer uptime guarantees. For example, how can Rosa demand a QoS that ensures all photos of dogs uploaded to her app guarantee the specific dog breeds are returned so that users can look up their other friend's 'border collie's? If dog breeds are returned, what ontologies exist for breeds? Are they consistent with each other, or shortened? ('Collie' versus 'border collie'; 'staffy' versus 'staffordshire bull terrier'?) For some applications, these unstated QoS metrics specific to the ML service may have significant legal ramifications.

2) Web service documentation and documenting ML: From the *developer's* perspective, little has been achieved to assess intelligent service quality or assure quality of these computer vision services. Web services and their interfaces (APIs) are the bridge between developers' needs and the software components [29]; therefore, assessing such computer vision services from the quality of their APIs is thereby directly related to the development quality [30]. Good APIs should be intuitive and require less documentation browsing [31], thereby increasing productivity. Conversely, poor APIs that are hard to understand and work with reduce developer productivity, thereby reducing product quality. This typically leads to developers congregating on forums such as Stack Overflow, leading to a repository of unstructured knowledge likely to concern API design [32]. The consequences of addressing these concerns in development leads to a higher demand in technical support (as measured in [33]) that, ultimately, causes the maintenance to be far more expensive, a phenomenon widely known in software engineering economics [34]. Rosa, for instance, isn't aware of technical ML concepts; if she cannot reason about what search results are relevant when browsing the service and understanding functionality, her productivity is significantly decreased. Conceptual understanding is critical for using APIs, as demonstrated by Ko and Riche, and the effects of maintenance this may have in the future of her application is unknown.

Recent attempts to document attributes and characteristics on ML models have been proposed. Model cards were introduced by Mitchell et al. [36] to describe how particular models were trained and benchmarked, thereby assisting users to reason if the model is right for their purposes and if it can achieve its stated outcomes. Gebru et al. [37] also proposed datasheets, a standardised documentation format to describe the need for a particular data set, the information contained within it and what scenarios it should be used for, including legal or ethical concerns.

However, while target audiences for these documents may be of a more technical AI level (i.e., the ML engineer), there is still no standardised communication format for application developers to reason about using particular intelligent services, and the ramifications this may have on the applications they write is not fully conveyed. Hence, our work is focused on the application developer perspective.

IV. Method

This study organically evolved by observing phenomena surrounding computer vision services by assessing both their documentation and responses. We adopted a mixed methods approach, performing both qualitative and quantitative data collection on these two key aspects by using documentary research methods for inspecting the documentation and structured observations to quantitatively analyse the results over time. This, ultimately, helped us shape the following research hypotheses which this paper addresses:

- **[RH1]** Computer vision services do not respond with consistent outputs between services, given the same input image.
- **[RH2]** The responses from computer vision services are non-deterministic and evolving, and the same service can change its top-most response over time given the same input image.
- **[RH3]** Computer vision services do not effectively communicate this evolution and instability, introducing risk into engineering these systems.

We conducted two experiments to address these hypotheses against three popular computer vision services: AWS Rekognition [43], Google Cloud Vision [41], Azure Computer Vision [42]. Specifically, we targeted the AWS `DetectLabels` endpoint [56], the Google Cloud Vision `annotate:images` endpoint [55] and Azure's `analyze` endpoint [57]. For the remainder of this paper, we de-identify our selected computer vision services by labelling them as services A, B and C but do not reveal mapping to prevent any implicit bias. Our selection criteria for using these particular three services are based on the weight behind each service provider given their prominence in the industry (Amazon, Google and Microsoft), the ubiquity of their hosting cloud platforms as industry leaders of cloud computing (i.e., AWS, Google Cloud and Azure), being in the top three most adopted cloud vendors in enterprise applications in 2018 [38] and the consistent popularity of discussion amongst developers in developer communities such as Stack Overflow. While we choose these particular cloud computer vision services, we acknowledge that similar services [45, 44, 47, 46, 48–50] also exist, including other popular services used in Asia [51–54] (some offering 3D image analysis [58]). We reflect on the impacts this has to our study design in section VII.

Our study involved an 11-month longitudinal study which consisted of two 13 week and 17 week experiments from April to August 2018 and November 2018 to March 2019, respectively. Our investigation into documentation occurred on August 28 2018. In total, we assessed the services with three data sets; we first ran a pilot study using a smaller pool of 30 images to confirm the end-points remain stable, re-running the study with a larger pool of images of 1,650 and 5,000 images. Our selection criteria for these three data sets were that the images had to have varying objects, taken in various scenes and various times. Images also needed to contain disparate objects. Our small data set was sourced by the first author by taking photos of random scenes in an afternoon, whilst our second data set was sourced from various members of our research group from their personal photo libraries. We also wanted to include a data set that was publicly available prior to running our study, so for this data set we chose the COCO 2017 validation data set [39]. We have made our other two data sets available online ([59]). We collected results and their responses from each service's API endpoint using a python script [60] that sent requests to each service periodically via cron jobs. Table I summarises various characteristics about the data sets used in these experiments.

TABLE I: Characteristics of our data sets and responses.

Data set	**Small** [59]	**Large** [59]	**COCOVal17** [39]
# Images/data set	30	1,650	5000
# Unique labels found	307	3506	4507
Number of snapshots	9	22	22
Avg. days b/n requests	12 Days	8 Days	8 Days

We then performed quantitative analyses on each response's labels, ensuring all labels were lowercased as case changed for services A and C over the evaluation period. To derive at the consistency of responses for each image, we considered only the 'top' labels per image for each service and data set. That is, for the same image i over all images in data set D where $i \in D$ and over the three services, the top labels per image (T_i) of all labels per image L_i (i.e., $T_i \subseteq L_i$) is that where the respective label's confidences are consistently the highest of all labels returned. Typically, the top labels returned is a set containing only one element—that is, only one unique label consistently returned with the highest label ($|T_i| = 1$)—however there are cases where the top labels contains multiple elements as their respective confidences are *equal* ($|T_i| > 1$).

We measure response consistency under 6 aspects:

1) **Consistency of the top label between each service.** Where the same image of, for example, a dog is sent to the three services, the top label for service A may be 'animal', B 'canine' and C 'animal'. Therefore, service B is inconsistent.
2) **Semantic consistency of the top labels.** Where a service has returned multiple top labels ($|T_i| > 1$), there may lie semantic differences in what the service thinks the image best represents. Therefore, there is conceptual inconsistency in the top labels for a service even when the confidences are equal.
3) **Consistency of the top label's confidence per service.** The top label for an image does not guarantee a high confidence. Therefore, there may be inconsistencies in how confident the top labels for all images in a service is.
4) **Consistency of confidence in the intersecting top label between each service.** The spread of a top intersecting

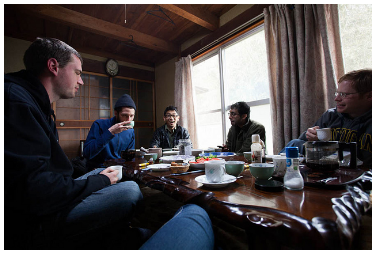


Fig. 1: The only consistent label for the above image is 'people' for services C and B. The top label for A is 'conversation' and this label is not registered amongst the other two services.

TABLE II: Ratio of the top labels (to images) that intersect in each data set for each permutation of service.

Service	Small	Large	COCOVal17	µ	σ
A ∩ B ∩ C	3.33%	2.73%	4.68%	2.75%	0.0100
A ∩ B	6.67%	11.27%	12.26%	10.07%	0.0299
A ∩ C	20.00%	13.94%	17.28%	17.07%	0.0304
B ∩ C	6.67%	12.97%	20.90%	13.51%	0.0713

label, e.g. 'cat', may not have the same confidences per service even when all three services agree that 'cat' is the top label. Therefore, there is inconsistency in the confidences of a top label even where all three services agree.

5) **Consistency of the top label over time.** Given an image, the top label in one week may differ from the top label the following week. Therefore, there is inconsistency in the top label itself due to model evolution.
6) **Consistency of the top label's confidence over time.** The top label of an image may remain static from one week to the next for the same service, but its confidence values may change with time. Therefore, there is inconsistency in the top label's confidence due to model evolution.

For the above aspects of consistency, we calculated the spread of variation for the top label's confidences of each service for every 1 percent point; that is, the frequency of top label confidences within 100–99%, 99–98% etc. The consistency of top label's and their confidences between each service was determined by intersecting the labels of each service per image and grouping the intersecting label's confidences together. This allowed us to determine relevant probability distributions. For reproducibility, all quantitative analysis is available online [61].

V. Findings

A. Consistency of top labels

1) Consistency across services: Table II presents the consistency of the top labels between data sets, as measured by the cardinality of the intersection of all three services' set of top labels divided by the number of images per data set. A combination of services present varied overlaps in their top labels; services A and C provide the best overlap for all three data sets, however the intersection of all three irrespective of data sets is low.

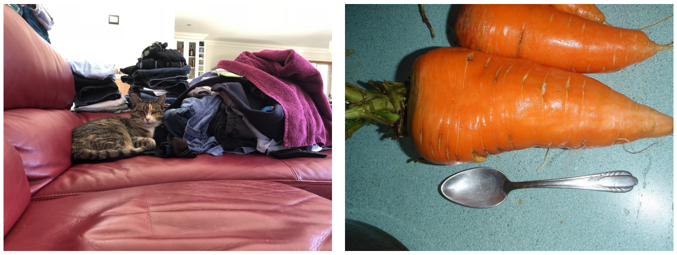

(a)
(b)

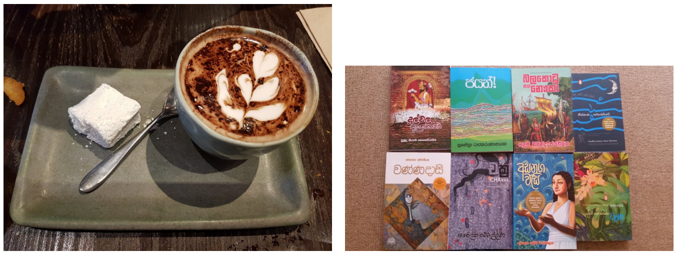
Fig. 2: *Left:* The top labels for each service do not intersect, with each having a varied ontology: T_i = { A = {'black'}, B = {'indoor'}, C = {'slide', 'toy'} }. (Service C returns *both* 'slide' and 'toy' with equal confidence.) *Right:* The top labels for each service focus on disparate subjects in the image: $T_i =$ { A = { 'carrot' }, B = { 'indoor' }, C = { 'spoon' } }.

The implication here is that, without semantic comparison (see section VII), service vendors are not 'plug-and-play'. If Rosa uploaded the sample images in this paper to her application to all services, she would find that only fig. 1 responds with 'person' for services B and C in their respective set of top labels. However, if she decides to then adopt service A, then fig. 1's top label becomes 'conversation'; the 'person' label does not appear within the top 15 labels for service A and, conversely, the 'conversation' label does not appear in the other services top 15.

Should she decide if the performance of a particular service isn't to her needs, then the vocabulary used for these labels becomes inconsistent for all other images; that is, the top label sets per service for fig. 2a shows no intersection at all. Furthermore, the part of the image each service focuses on may not be consistent for their top labels; in fig. 2b, service A's top label focuses on the vegetable ('carrot'), service C focuses on the 'spoon', while service B's focus is that the image is 'indoor's. It is interesting to note that service B focuses on the scene matter (indoors) rather than the subject matter. (Furthermore, we do not actually know if the image in fig. 2b was taken indoors.)

Hence, developers should ensure that the vocabulary used by a particular service is right for them before implementation. As each service does not work to the same standardised model, trained with disparate training data, and tuned differently, results will differ despite the same input. This is unlike deterministic systems: for example, switching from AWS Object Storage to Google Cloud Object storage will conceptually provide the same output (storing files) for the same input (uploading files). However, computer vision services do not agree on the top label for images, and therefore developers are likely to be vendor locked, making changes between services non-trivial.

2) Semantic consistency where $|T_i| > 1$: Service C returns two top labels for fig. 2a; 'slide' and 'toy'. More than one



(a)

(b)

Fig. 3: *Left:* Service C is 98.49% confident of the following labels: { 'beverage', 'chocolate', 'cup', 'dessert', 'drink', 'food', 'hot chocolate' }. However, it is up to the developer to decide which label to persist with as all are returned. *Right:* Service B persistently returns a top label set of { 'book', 'several' }. Both are semantically correct for the image, but disparate in what the label is to describe.

top label is typically returned in service C (80.00%, 56.97%, and 81.66% of all images for all three data sets, respectively) though this also occurs in B in the large (4.97% of all images) and COCOVal17 data sets (2.38%). Semantic inconsistencies of what this label conceptually represents becomes a concern as these labels have confidences of *equal highest* consistency. Thus, some services are inconsistent in themselves and cannot give a guaranteed answer of what exists in an image; services C and B have multiple top labels, but the respective services cannot 'agree' on what the top label actually is. In fig. 3a, service C presents a reasonably high confidence for the set of 7 top labels it returns, however there is too much diversity ranging from a 'hot chocolate' to the hypernym 'food'. Both are technically correct, but it is up to the developer to decide the level of hypernymy to label the image as. We also observe a similar effect in fig. 3b, where the image is labelled with both the subject matter and the number of subjects per image.

Thus, a taxonomy of ontologies is unknown; if a 'border collie' is detected in an image, does this imply the hypernym 'dog' is detected, and then 'mammal', then 'animal', then 'object'? Only service B documents a taxonomy for capturing what level of scope is desired, providing what it calls the '86-category' concept as found in its how-to guide:

> *"Identify and categorize an entire image, using a category taxonomy with parent/child hereditary hierarchies. Categories can be used alone, or with our new tagging models."* [62]

Thus, even if Rosa implemented conceptual similarity analysis for the image, the top label set may not provide sufficient information to derive at a conclusive answer, and if simply relying on only one label in this set, information such as the duplicity of objects (e.g., 'several' in fig. 3b) may be missed.

B. Consistency of confidence

1) Consistency of top label's confidence: In fig. 4, we see that there is high probability that top labels have high confidences for all services. In summary, one in nine images

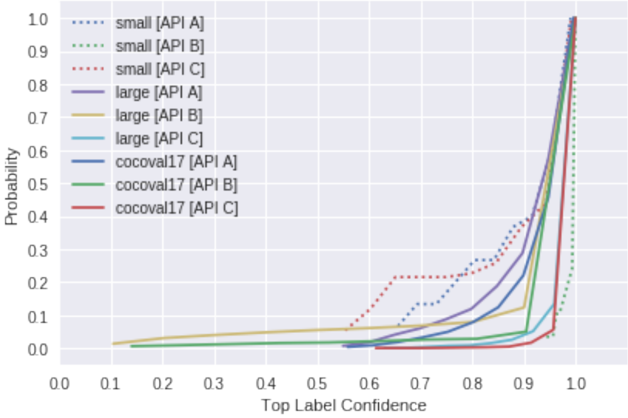


Fig. 4: Cumulative distribution of the top labels' confidences. One in nine images return a top label(s) confident to $\gtrsim$ 97%, though there is a wider distribution for service A.

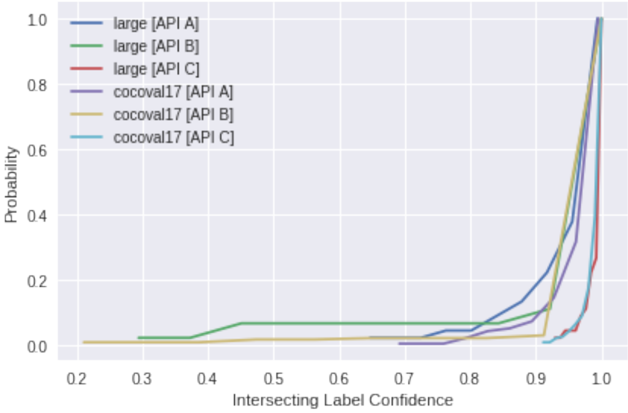


Fig. 5: Cumulative distribution of intersecting labels top labels' confidences. The small data set is intentionally removed due to low intersections of labels (see table II).

uploaded to any service will return a top label confident to at least 97%. However, there is higher probability for service A returning a lower confidence, followed by B. The best performing service is C, with 90% of requests having a top label confident to $\gtrsim$ 95%, when compared to $\gtrsim$ 87% and $\gtrsim$ 93% for services A and B, respectively.

Therefore, Rosa could generally expect that the top labels she receives in her images do have high confidence. That is, each service will return a top label that they are confident about. This result is expected, considering that the 'top' label is measured by the highest confidence, though it is interesting to note that some services are generally more confident than others in what they present back to users.

2) Consistency of intersecting top label's confidence: Even where all three services do agree on a set of top labels, the disparity of how much they agree by is still of importance. Just because three services agree that an image contains consistent top labels, they do not always have a small spread of confidence. In fig. 6, the three services agree

TABLE III: Ratio of the top labels (to images) that remained the top label but changed confidence values between intervals.

Service	Small	Large	COCOVal17	$\mu(\delta_c)$	$\sigma(\delta_c)$	Median(δ_c)	Range(δ_c)
A	53.33%	59.19%	44.92%	9.62e−8	6.84e−8	5.96e−8	[5.96e−8, 6.56e−7]
B	0.00%	0.00%	0.02%	-	-	-	-
C	33.33%	41.36%	15.60%	5.35e−7	8.76e−7	3.05e−7	[1.27e−7, 1.13e−5]

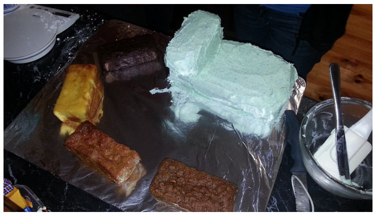


Fig. 6: All three services agree the top label for the above image is ‘food’, but the confidences to which they agree by vary significantly. Service C is most confident to 94.93% (in addition with the label ‘bread’); service A is the second most confident to 84.32%; service B is the least confident with 41.39%.

with $\sigma = 0.277$, significantly larger than that of all images in general $\sigma = 0.0831$. Figure 5 displays the cumulative distribution of all intersecting top labels’ confidence values, presenting slightly similar results to that of fig. 4.

C. Evolution risk

1) Label Stability: Generally, the top label(s) did not evolve in the evaluation period. 16.19% and 5.85% of images did change their top label(s) in the Large and COCOVal17 data sets in service A. Thus, top labels are stable but not guaranteed to be constant.

2) Confidence Stability: Similarly, where the top label(s) remained the same from one interval to the next, the confidence values were stable. Table III displays the proportion of images that changed their top label’s confidence values with various statistics on the confidence deltas between snapshots (δ_c). However, this delta is so minuscule that we attribute such changes to statistical noise.

VI. Recommendations

A. Recommendations for intelligent service users

1) Test with a representative ontology for the particular use case: Rosa should ensure that in her testing strategies for the app she develops, there is an ontology focus for the types of vocabulary that are returned. Additionally, we noted that there was a sudden change in case for services A and C; for all comparative purposes of labels, each label should be lower-cased.

2) Incorporate a specialised intelligent service testing methodology into the development lifecycle: Rosa can utilise the different aspects of consistency as outlined in this paper as part of her quality strategy. To ensure results are correct over time, we recommend developers create a representative data set of the intended application’s data set and evaluate these changes against their chosen service frequently. This will help identify when changes, if any, have occurred if vendors do not provide a line of communication when this occurs.

3) Intelligent services are not ‘plug-and-play’: Rosa will be locked into whichever vendor she chooses as there is inherent inconsistency between these services in both the vocabulary and ontologies that they use. We have demonstrated that very few services overlap in their vocabularies, chiefly because they are still in early development and there is yet to be an established, standardised vocabulary that can be shared amongst the different vendors. Issues such as those shown in section V-A can therefore be avoided.

Throughout this work, we observed that the terminologies used by the various vendors are different. Documentation was studied, and we note that there is inconsistency between the ways techniques are described to users. We note the disparity between the terms ‘detection’, ‘recognition’, ‘localisation’ and ‘analysis’. This applies chiefly to object- and facial-related techniques. Detection applies to facial detection, which gives bounding box coordinates around all faces in an image. Similarly, localisation applies the same methodology to disparate objects in an image and labels them. In the context of facial ‘recognition’, this term implies that a face is *recognised* against a known set of faces. Lastly, ‘analysis’ applies in the context of facial analysis (gender, eye colour, expression etc.); there does not exist a similar analysis technique on objects.

We notice similar patterns with object ‘tagging’, ‘detection’ and ‘labelling’. Service A uses ‘Entity Detection’ for object categorisation, service B uses ‘Image Tagging’, and service C uses the term ‘Detect Labels’ : conceptually, these provide the same functionality but the lack of consistency used between all three providers is concerning and leaves room for confusion with developers during any comparative analyses. Rosa may find that she wants to label her images into day/night scenes, but this in turn means the ‘labelling’ of varying objects. There is therefore no consistent standards to use the same terminology for the same concepts, as there are in other developer areas (such as Web Development).

4) Avoid use in safety-critical systems: We have demonstrated in this paper that both labels and confidences are stable but not constant; there is still an evolution risk posed to developers that may cause unknown consequences in applications dependent on these computer vision services. Developers should avoid their use in safety critical systems due to the lack of visible changes.

B. Recommendations for intelligent service providers

1) Improve the documentation: Rosa does not know that service A returns back ‘carrot’ for its top response, with service C returning ‘spoon’ (fig. 2b). She is unable to tell the service’s API where to focus on the image. Moreover, how can she toggle the level of specificity in her results? She is frustrated that service C can detect ‘chocolate’, ‘food’ and also ‘beverage’ all as the same top label in fig. 3a: what label is she to choose when the service is meant to do so for her, and how does she get around this? Thus, we recommend vendors to improve the documentation of services by making known the boundary set of the training data used for the algorithms. By making such information publicly available, developers would be able to review the service’s specificity for their intended use case (e.g., maybe Rosa is satisfied her app can catalogue ‘food’ together, and in fact does not want specific types of foods (‘hot chocolate’) catalogued). We also recommend that vendors publish usage guidelines should that include details of priors and how to evaluate the specific service results.

Furthermore, we did not observe that the vendors documented how some images may respond with multiple labels of the exact same confidence value. It is not clear from the documentation that response objects can have duplicate top values, and tutorials and examples provided by the vendors do not consider this possibility. It is therefore left to the developer to decide which label from this top set of labels best suits for their particular use case; the documentation should describe that a rule engine may need to be added in the developer’s application to verify responses. The implications this would have on maintenance would be significant.

2) Improve versioning: We recommend introducing a versioning system so that a model can be used from a specific date in production systems: when Rosa tests her app today, she would like the service to remain *static* the same for when her app is deployed in production tomorrow. Thus, in a request made to the vendor, Rosa could specify what date she ran her app’s QA testing on so that she knows that henceforth these model changes will not affect her app.

3) Improve Metadata in Response: Much of the information in these services is reduced to a single confidence value within the response object, and the details about training data and the internal AI architecture remains unknown; little metadata is provided back to developers that encompass such detail. Early work into model cards and datasheets [36, 37] suggests more can be done to document attributes about ML systems, however at a minimum from our work, we recommend including a reference point via the form of an additional identifier. This identifier must also permit the developers to submit the identifier to another API endpoint should the developer wish to find further characteristics about the AI empowering the intelligent service, reinforcing the need for those presented in model cards and datasheets. For example, if Rosa sends this identifier she receives in the response object to the intelligent service descriptor API, she could find out additional information such as the version number or date when the model was trained, thereby resolving potential evolution risk, and/or the ontology of labels.

4) Apply constraints for predictions on all inputs: In this study, we used some images with intentionally disparate, and noisy objects. If services are not fully confident in the responses they give back, a form of customised error message should be returned. For example, if Rosa uploads an image of 10 various objects on a table, rather than returning a list of top labels with varying confidences, it may be best to return a ‘too many objects’ exception. Similarly, if Rosa uploads a photo that the model has had no priors on, it might be useful to return an ‘unknown object’ exception than to return a label it has no confidence of. We do however acknowledge that current state of the art computer vision techniques may have limits in what they can and cannot detect, but this limitation can be exposed in the documentation to the developers.

A further example is sending a one pixel image to the service, analogous to sending an empty file. When we uploaded a single pixel white image to service A, we received responses such as ‘microwave oven’, ‘text’, ‘sky’, ‘white’ and ‘black’ with confidences ranging from 51–95%. Prior checks should be performed on all input data, returning an ‘insufficient information’ error where any input data is below the information of its training data.

VII. Threats to Validity

A. Internal Validity

Not all computer vision services were assessed. As suggested in section IV, we note that there are other computer vision services such as IBM Watson. Many services from Asia were also not considered due to language barriers (of the authors) in assessing these services. We limited our study to the most popular three providers (outside of Asia) to maintain focus in this body of work.

A custom confidence threshold was not set. All responses returned from each of the services were included for analysis; where confidences were low, they were still included for analysis. This is because we used the default thresholds of each API to hint at what real-world applications may be like when testing and evaluating these services.

The label string returned from each service was only considered. It is common for some labels to respond back that are conceptually similar (e.g., ‘car’ vs. ‘automobile’) or grammatically different (e.g., ‘clothes’ vs. ‘clothing’). While we could have employed more conceptual comparison or grammatical fixes in this study, we chose only to compare lowercased labels and as returned. We leave semantic comparison open to future work.

Only introductory analysis has been applied in assessing the documentation of these services. Further detailed analysis of documentation quality against a rigorous documentation quality framework would be needed to fortify our analysis of the evolution of these services’ documentation.

B. External Validity

The documentation and services do change over time and evolve, with many allowing for contributions from the developer community via GitHub. We note that our evaluation of the documentation was conducted on a single date (see section IV) and acknowledge that the documentation may have changed from the evaluation date to the time of this publication. We also acknowledge that the responses and labelling may have evolved too since the evaluation period described and the date of this publication. Thus, this may have an impact on the results we have produced in this paper compared to current, real-world results. To mitigate this, we have supplied the raw responses available online [63].

Moreover, in this paper we have investigated *computer vision* services. Thus, the significance of our results to other domains such as natural language processing or audio transcription is, therefore, unknown. Future studies may wish to repeat our methodology on other domains to validate if similar patterns occur; we remain this open for future work.

C. Construct Validity

It is not clear if all the recommendations proposed in section VI are feasible or implementable in practice. Construct validity defines how well an experiment measures up to its claims; the experiments proposed in this paper support our three hypotheses but these have been conducted in a clinical condition. Real-world case studies and feedback from developers and providers in industry would remove the controlled nature of our work.

VIII. Conclusions & Future Work

This study explored three popular computer vision services over an 11 month longitudinal experiment to determine if these services pose any evolution risk or inconsistency. We find that these services are generally stable but behave inconsistently; responses from these services do change with time and this is not visible to the developers who use them. Furthermore, the limitations of these systems are not properly conveyed by vendors. From our analysis, we present a set of recommendations for both intelligent service vendors and developers.

Standardised software quality models (e.g., [11]) target maintainability and reliability as primary characteristics. Quality software is stable, testable, fault tolerant, easy to change and mature. These computer vision services are, however, in a nascent stage, difficult to evaluate, and currently are not easily interchangeable. Effectively, the intelligent service response objects are shifting in material ways to developers, albeit slowly, and vendors do not communicate this evolution or modify API endpoints; the endpoint remains static but the content returned does not despite the same input.

There are many potential directions stemming from this work. To start, we plan to focus on preparing a more comprehensive datasheet specifically targeted at what should be documented to application developers, and not data scientists. Reapplying this work in real-world contexts, that is, to get real developer opinions and study production grade systems, would also be beneficial to understand these phenomena in-context. This will help us clarify if such changes are a real concern for developers (i.e., if they really need to change between services, or the service evolution has real impact on their applications). We also wish to refine and systematise the method used in this study and develop change detectors that can be used to identify evolution in these services that can be applied to specific ML domains (i.e., not just computer vision), data sets, and API endpoints, thereby assisting application developers in their testing strategies. Moreover, future studies may wish to expand the methodology applied by refining how the responses are compared. As there does not yet exist a standardised list of terms available between services, labels could be *semantically* compared instead of using exact matches (e.g., by using stem words and synonyms to compare similar meanings of these labels), similar to previous studies [40].

This paper has highlighted only some high-level issues that may be involved in using these evolving services. The laws of software evolution suggest that for software to be useful, it must evolve [10, 9]. There is, therefore, a trade-off, as we have shown, between consistency and evolution in this space. For a component to be stable, any changes to dependencies it relies on must be communicated. We are yet to see this maturity of communication from intelligent service providers. Thus, developers must be cautious between integrating intelligent components into their applications at the expense of stability; as the field of AI is moving quickly, we are more likely to see further instability and evolution in intelligent services as a consequence.

References

[1] A. Reis, D. Paulino, V. Filipe, and J. Barroso, "Using Online Artificial Vision Services to Assist the Blind - an Assessment of Microsoft Cognitive Services and Google Cloud Vision." *WorldCIST*, vol. 746, no. 12, pp. 174–184, 2018.

[2] H. da Mota Silveira and L. C. Martini, "How the New Approaches on Cloud Computer Vision can Contribute to Growth of Assistive Technologies to Visually Impaired in the Following Years," *Journal of Information Systems Engineering and Management*, vol. 2, no. 2, pp. 1–3, 2017.

[3] R. S. Pressman, *Software engineering: a practitioner's approach.* Palgrave Macmillan, 2005.

[4] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2011.

[5] J. W. Horch, *Practical guide to software quality management.* Artech House, 2003.

[6] S. Demeyer and T. Mens, *Software Evolution.* Springer, 2008.

[7] M. W. Godfrey and D. M. German, "The past, present, and future of software evolution," in *2008 Frontiers of Software Maintenance*, Sep. 2008, pp. 129–138.

[8] Q. Tu *et al.*, "Evolution in open source software: A case study," in *Proceedings 2000 International Conference on Software Maintenance.* IEEE, 2000, pp. 131–142.

[9] T. Mens, M. Wermelinger, S. Ducasse, S. Demeyer, R. Hirschfeld, and M. Jazayeri, "Challenges in software evolution," in *Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*, Sep. 2005, pp. 13–22.

[10] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, "Studying software evolution using topic models," *Science of Computer Programming*, vol. 80, pp. 457 – 479, 2014.

[11] International Organization for Standardization, "**Information technology – Software product quality**," Nov. 1999.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[13] H. Hosseini, B. Xiao, and R. Poovendran, "Google's Cloud Vision API is Not Robust to Noise," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Jan. 2018, pp. 101–105.

[14] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, "The elephant in the room," *arXiv preprint arXiv:1808.03305*, 2018.

[15] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, "With Great Training Comes Great Vulnerability - Practical Attacks against Transfer Learning." *USENIX Security Symposium*, 2018.

[16] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks." *IEEE Transactions on Evolutionary Computation*, 2019.

[17] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust Physical-World Attacks on Deep Learning Visual Classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, vol. cs.CV, 2016.

[19] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust Physical-World Attacks on Deep Learning Visual Classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[20] Z. Pezzementi, T. Tabor, S. Yim, J. K. Chang, B. Drozd, D. Guttendorf, M. Wagner, and P. Koopman, "Putting image manipulations in context: robustness testing for safe perception," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2018, pp. 1–8.

[21] A. Arpteg, B. Brinne, L. Crnkovic-Friis, and J. Bosch, "Software Engineering Challenges of Deep Learning," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, Oct. 2018, pp. 50–59.

[22] C. Murphy, G. E. Kaiser, and M. Arias, "An approach to software testing of machine learning applications," 2007.

[23] H. B. Braiek and F. Khomh, "On testing machine learning programs," *arXiv preprint arXiv:1812.02257*, 2018.

[24] Y. Nishi, S. Masuda, H. Ogawa, and K. Uetsuki, "A test architecture for machine learning product," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2018, pp. 273–278.

[25] C. Murphy and G. E. Kaiser, "Improving the dependability of machine learning applications," 2008.

[26] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "Whats your ML Test Score? A rubric for ML production systems," 2016.

[27] A. L. M. Ortiz, "Curating Content with Google Machine Learning Application Programming Interfaces," in *EIA-Portugal*, Jul. 2017.

[28] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A Framework for Comparing and Ranking Cloud Services," in *2011 IEEE 4th International Conference on Utility and Cloud Computing (UCC 2011)*. IEEE, Nov. 2011, pp. 210–218.

[29] K. Arnold, "Programmers are people, too," *ACM Queue*, vol. 3, no. 5, pp. 54–59, 2005.

[30] A. J. Ko, B. A. Myers, and H. H. Aung, "Six learning barriers in end-user programming systems," in *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE, 2004, pp. 199–206.

[31] M. Piccioni, C. A. Furia, and B. Meyer, "An Empirical Study of API Usability," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 5–14.

[32] W. Wang, H. Malik, and M. W. Godfrey, "Recommending Posts concerning API Issues in Developer Q&A Sites," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, May 2015, pp. 224–234.

[33] M. Henning, "API design matters," *Commun. ACM*, vol. 52, no. 5, pp. 46–56, 2009.

[34] B. W. Boehm, *Software engineering economics*. Prentice-hall Englewood Cliffs (NJ), 1981, vol. 197.

[35] A. J. Ko and Y. Riche, "The role of conceptual knowledge in API usability," in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2011, pp. 173–176.

[36] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," *arXiv preprint arXiv:1810.03993*, pp. 220–229, 2018.

[37] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumeé III, and K. Crawford, "Datasheets for datasets," *arXiv preprint arXiv:1803.09010*, pp. 1–17, 2018.

[38] RightScale Inc., "RightScale 2018 State of the Cloud Report," Tech. Rep., 2018.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[40] T. Ohtake, A. Cummaudo, M. Abdelrazek, R. Vasa, and

J. Grundy, “Merging Intelligent API Responses using a Proportional Representation Approach,” in *International Conference on Web Engineering ICWE*, Jun. 2019, pp. 391–406.

Online Artefacts

[41] “Vision API - Image Content Analysis — Cloud Vision API — Google Cloud,” http://bit.ly/2TD9mBs, accessed: 13 September 2018.

[42] “Image Processing with the Computer Vision API — Microsoft Azure,” http://bit.ly/2YqhkS6, accessed: 13 September 2018.

[43] “Amazon Rekognition,” https://amzn.to/2TyT2BL, accessed: 13 September 2018.

[44] “Detecting labels in an image,” http://bit.ly/2UlkW9K, accessed: 13 September 2018.

[45] “Watson visual recognition,” https://ibm.co/2TBNIO4, accessed: 13 September 2018.

[46] “Image Recognition API & Visual Search Results,” http://bit.ly/2UmNPCw, accessed: 13 September 2018.

[47] “Clarifai,” http://bit.ly/2TB3kSa, accessed: 13 September 2018.

[48] “Image Recognition API — DeepAI,” http://bit.ly/2TBNYgf, accessed: 26 September 2018.

[49] “Imagga - powerful image recognition apis for automated categorization & tagging in the cloud and on-premises,” http://bit.ly/2TxsyRe, accessed: 13 September 2018.

[50] “Image recognition - talkwalker,” http://bit.ly/2TyT7W5, accessed: 13 September 2018.

[51] “Megvii,” http://bit.ly/2WJYFzk, accessed: 3 April 2019.

[52] “Tuputech,” http://bit.ly/2uF4IsN, accessed: 3 April 2019.

[53] “Yitu technology,” http://bit.ly/2uGvxgf, accessed: 3 April 2019.

[54] “Sensetime,” http://bit.ly/2WH6RjF, accessed: 3 April 2019.

[55] “Detect Labels — Google Cloud Vision API Documentation — Google Cloud,” http://bit.ly/2TD5kcy, accessed: 28 August 2018.

[56] “Detecting labels in an image,” https://amzn.to/2TBNtTa, accessed: 28 August 2018.

[57] “How to call the Computer Vision API,” http://bit.ly/2TD5oJk, accessed: 28 August 2018.

[58] “Deepglint,” http://bit.ly/2uHHdPS, accessed: 3 April 2019.

[59] http://bit.ly/2KlyhcD, accessed: 27 March 2019.

[60] http://bit.ly/2G6ZOeC, accessed: 27 March 2019.

[61] http://bit.ly/2G7saFJ, note = Accessed: 27 March 2019.

[62] “What is Computer Vision?” http://bit.ly/2TDgUnU, accessed: 28 August 2018.

[63] http://bit.ly/2G5ZEEe, accessed: 27 March 2019.