# An Approach for Automatic and Large Scale Image Forensics

Thamme Gowda[1,2], Kyle Hundman[2], Chris A. Mattmann[1,2]
thammegowda.n@usc.edu

[1]Computer Science Department
University of Southern California
Los Angeles, CA 90089 USA

[2]Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109 USA

## ABSTRACT

This paper describes the applications of deep learning-based image recognition in the DARPA Memex program and its repository of 1.4 million weapons-related images collected from the Deep web. We develop a fast, efficient, and easily deployable framework for integrating Google's Tensorflow framework with Apache Tika for automatically performing image forensics on the Memex data. Our framework and its integration are evaluated qualitatively and quantitatively and our work suggests that automated, large-scale, and reliable image classification and forensics can be widely used and deployed in bulk analysis for answering domain-specific questions.

## CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Image search**;
• **Applied computing** → *Computer forensics*;

## KEYWORDS

Image Recognition, Multimedia Forensics, Information Retrieval

**ACM Reference format:**
Thamme Gowda[1,2], Kyle Hundman[2], Chris A. Mattmann[1,2]. 2017. An Approach for Automatic and Large Scale Image Forensics. In *Proceedings of MFSec'17, Bucharest, Romania, June 06, 2017,* 5 pages.
https://doi.org/http://dx.doi.org/10.1145/3078897.3080536

## 1 INTRODUCTION

Over the past two years, our research team has borne witness to the ease and availability of potentially criminal goods and services on the modern Internet. In particular, our team's work on the DARPA Memex project has focused on the issues of online gun sales, as such sales can have grim consequences in that they provide a medium for buyers and sellers to circumvent traditional background checks. In turn, this proliferates the sale of dangerous semi-automatic weapons and can lead directly to loss of human life. For instance, a New York Police Department (NYPD) investigation in 2013 identified guns used in one suicide and four murders and traced their origin to transactions on the website armslist.com[16].


*MFSec'17, June 06, 2017, Bucharest, Romania*

ACM ISBN 978-1-4503-5034-1/17/06…$15.00
https://doi.org/http://dx.doi.org/10.1145/3078897.3080536

The ability to rapidly and automatically monitor these types of gun transactions is a significant challenge. In addition to Armslist, there are hundreds of both national and regional gun sales sites like floridaguntrader.com or gunbroker.com. And though the ads, like the whole Internet, contain a large amount of text [15], the proliferation of images necessitates object recognition and image analysis at scale. This is especially important because the actual content required to answer significant questions regarding these weapons ("is this an automatic weapon?", "is this a long or a short gun?", "are there multiple weapons being sold?") are the weapon *images* and not the ad text. Our recent work in DARPA's Memex initiative has expanded Apache Tika, a content detection and analysis framework [10], to support such analysis.

We have previously worked on bulk image analysis from the Deep web as it relates to human trafficking data [9] using the Apache Tika. However, that work focused on image metadata forensics as an alternative to image-pixel based analyses and object detection and recognition. Though metadata forensics were promising in human trafficking, weapons required pixel-based analyses. Based on our study of over 80 websites and online forums that specialize in the exchange of weapons, object recognition and computer vision were needed to automatically discern whether or not the guns being sold are automatic or semi-automatic, whether they have been stolen (using serial-number identification), and whether the transactions are potentially illegal. Automatically being able to discern these types of object properties in bulk analyses of image data has the potential to thwart crimes and, ultimately, to save lives.

Historically, the best object recognition systems were inaccurate, but this has changed due to recent advancements in deep neural networks, larger training datasets, and improved computing resources. Tensorflow is a scalable, Python-based system and it natively supports image recognition via its *Inception* model [1]. *Inception* provides a neural network trained on the ImageNet corpus [7], a dataset of 14,197,122 images classified using text from the WordNet taxonomy. The integration of Tensorflow with Tika batch processing methods such as Tika Spark now enables Tensorflow processing to be parallelized. The end result is a highly-scalable, off-the-shelf system that can accurately identify and classify objects in images into a thousand categories. This capability – combined with Apache Tika's native support for detecting thousands of file formats and extracting their metadata and textual content – is an attractive, automated solution that can perform bulk analysis in the weapons domain, but more generally, in any context where text and images are present and such analyses are required.

Despite its merit, this integration of Tensorflow with Tika presented a significant challenge: Tensorflow does not provide default

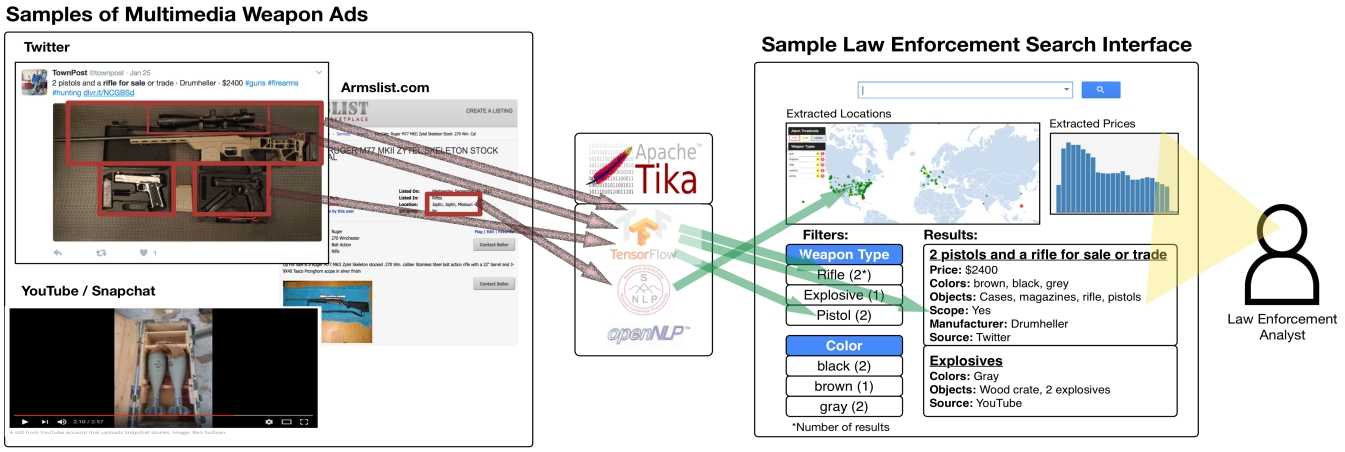


**Figure 1: This diagram demonstrates how the integration of Tika and Tensorflow facilitates in-depth search across heterogeneous content types. There are several extensions fo our object recognition implementation as well, including more refined categories, optical-character recognition, and image similarity metrics.**

bindings to Java-based frameworks. Apache Tika is primarily written in Java and thus integrating with Tensorflow is not straightforward like with other JVM-compatible libraries. Our research directly addresses this and contributes several methods that make Tensorflow easier to integrate into Java-based systems like Tika, and any digital forensics system that can make a call to an application programming interface (API). In this paper, we report on our integration of Tika and Tensorflow using the weapons domain as a motivating example. We also evaluate the integration in both its robustness in object recognition without training beyond that of ImageNet. Lastly, we demonstrate that Tensorflow and Tika together form a scalable forensics solution for bulk Deep web image analysis.

## 2 DATA COLLECTION AND CONTENT ANALYSIS IN MEMEX

Commercial search engines provide generalized search interfaces that allow users to search across a limited portion of the web [14]. However, in the context of cyber security and law enforcement, commercial search engines miss essential content from the Deep and Dark web. Because it is hard to reach, this content often harbors illicit activity. The goal of Memex is to develop software that can quickly and thoroughly collect, organize, and search subsets of information relevant to individual domains of interest.

The program's initial focus was on human trafficking and the trade and sale of illicit goods, and several enhanced web crawlers were used to discover and retrieve information from the websites related to these domains. Along with the general-purpose web crawlers, specialized crawlers were used for the retrieval of Dark web data using The Onion Router (TOR) protocol [11] and also specialized in the retrieval of dynamic AJAX content guarded by login forms. Fetched data were then cached within the system for analysis due to the ephemeral nature of the source (web) content.

The Memex data included 7.2 million items of content in the illegal weapon sales domain, of which 1.4 million objects were images. Before processing images, we analyzed the textual documents in a separate experiment using named-entity recognition (NER) models that extracted people, locations, organizations, weapon names, and weapon types. Tika has recently added support for this task using popular natural language processing (NLP) toolkits like Stanford CoreNLP[3], Apache OpenNLP[12], and MIT Lincoln Lab's MITIE [13]. However, as we described in Section 1, our work has focused on weapons images for two primary reasons: (1) web crawlers generally extract any linked content from a site and ensuring that images contain relevant objects was an important preprocessing step, and (2) classifying image objects allows for the cataloguing of specific objects of interest. With regard to number two, the queries we sought to answer were related to automated identification of (semi-)automatic weapons and illegal gun transactions – this often requires direct analysis of the image rather than associated ad text.

The ultimate goal of integrating and characterizing diverse content scattered across the web is to provide law enforcement analysts with tools that will help them quickly identify potentially illegal activity, and images and videos often provide salient information not available in text; in many cases, illegal weapons dealers intentionally embed revealing details in rich content mediums because they are harder to identify. The integration of Tensorflow and Tika provides a single, streamlined platform that unites the extraction of textual and rich content. This combined content can then be exposed through search and visualization interfaces that improve analysts' abilities to drill-down and explore comprehensive, diverse content contained in weapons ads (see Figure 1).

## 3 INTEGRATION

To integrate Tika and Tensorflow, we extended Tika's `Recogniser` interface which was introduced as part of our work in integrating named-entity recognition (NER) toolkits described in the prior section. Our new interface was called `ObjectRecogniser`. The goal of `ObjectRecogniser` is to facilitate multiple implementations that extend beyond Tensorflow and may include other deep learning and object recognition frameworks in the future with additional effort. The main component of this interface contract is a function that accepts image data and returns a list of `RecognisedObjects`.

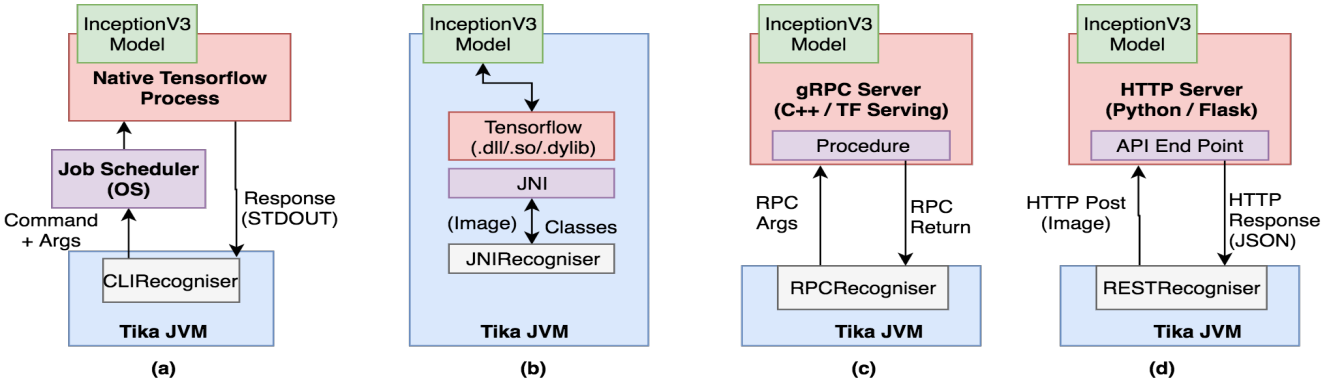


**Figure 2: Tika and Tensorflow Integration**

For our initial implementation of `ObjectRecogniser`, we created a Python-based command line (CLI) tool as an entry point to Tensorflow's image recognition network. Apache Tika executed in the JVM process where a new native process was created and destroyed. Tika then passed the image path as a command line argument to the tool as shown in Figure 2 (a). The tool parsed the arguments, passed the content to the Tensorflow network, and reported the results by printing it to standard output. Tika's parser then reads the result from its output stream. We did not extend Tensorflow's existing ImageNet/Inception model training and simply used it off-the-shelf and pre-configured in the Tensorflow Python program.

Vendors recommend the Java Native Interface (JNI) for integrating native code libraries to Java frameworks[4]. JNI acts as glue between bytecode instructions that run within the Java Virtual Machine (JVM) and the native code instructions that run directly on the CPU. At runtime, the bytecode of Tika (caller) and native code of Tensorflow (callee) runs within a single process from the operating system's perspective as shown in Figure 2 (b). Theoretically, this is the best way of merging the JVM world with native code, however the merit acquired in terms of the qualitative and quantitative metrics were not in our case worth the efforts.

The developers of the Tensorflow framework recommended using gRPC-based integration for the production systems[17]. gRPC is a client-server based architecture in which caller acts as an RPC client and callee operates as a server in a different address space. Unlike traditional RPC frameworks, gRPC is a high-performance, high-CPU, and bandwidth-efficient transport on top of HTTP/2 that supports full duplex streaming[6]. In our case, we embedded gRPC client in Tika JVM and exported Tensorflow image recognition capabilities as remote procedures via gRPC service. We used Tensorflow Serving, a gRPC server implemented in C++, and also created a Docker container to host it. Collecting the needed libraries to build the gRPC interface proved to be a non-trivial effort, and rather than require users of our Tika and Tensorflow integration to install these libraries, we also investigated building a Representation State Transfer (REST) interface [2].

REST is a client-server architecture paradigm for connecting heterogeneous systems without the need for states [2, Chapter 5]. The REST application programming interfaces (API) is powered by the HyperText Transfer Protocol (HTTP) which abstracts the complexities of Transmission Control Protocol. We created a REST API for Tensorflow image recognition using Python Flask. The Flask-based HTTP service registered a TCP port and offered HTTP API endpoints as shown in Figure 2 (d), and the REST interface had the advantage of minimizing client dependencies for using our framework. REST clients are lightweight and have easily installable dependencies across all major programming languages. Our REST API endpoint accepted HTTP POST requests with image data in the request body. This service loaded the *Inception v3* [19] model during the initialization phase and held the model in memory for reuse during the future HTTP Requests.

In the client side, Tika used `TensorflowRESTRecogniser` – an implementation based on HTTPClient – to transfer image content as HTTP Post request. The client parsed the JSON response from REST API to retrieve the object names, IDs and confidence scores. We also created a Docker specification for bootstrapping the Tensorflow image recognition REST API for semi-automated deployment of the system. This presents a user-friendly client and server for Tensorflow and Tika integration in which all needed dependencies and capabilities for the integration are automatically provided.

## 4 EVALUATION

We carefully evaluated our integrations in several qualitative and quantitative areas. For each of our integration interfaces (CLI, gRPC, and REST API), we ran the integration against a single test image from Wikipedia and measured the amount of time from passing the image to Tika (and Tensorflow) to the return of a classification result from the API. All single-image tests were run on an Ubuntu 14.04 LTS Docker container running on MacBook Pro 2013 model (2.8GhZ Core i7 and SSD storage) for test images of size 1024x768 pixels. The slowest integration by far was the CLI integration which took 3 seconds to return a result, the fastest integration was the REST integration at 253ms, about twice as fast as the gRPC integration at 598ms. However, REST uses more bandwidth due to additional meta-data introduced by HTTP headers in the packets, compared to gRPC which does not require additional HTTP headers.

The REST integration benefited from a pre-loaded *Inception v3* model, along with lightweight dependencies and low overhead. The gRPC interface, while fast, suffered qualitatively from relying on conflicting HTTP client library transitive dependencies, making it difficult to integrate with some functionality from older HTTP

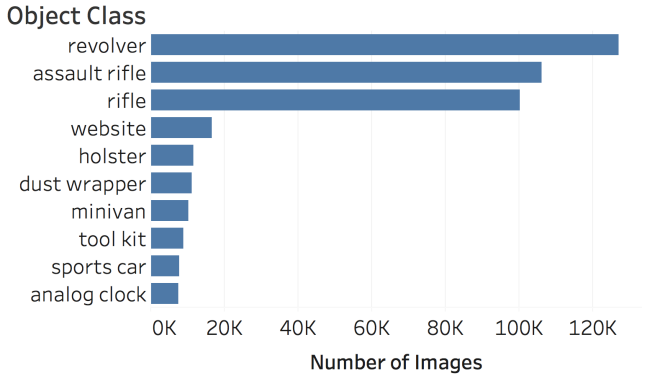


**Figure 3: Top 10 image classes found in the dataset. As more labeled weapon images become available, we will be able to train a custom classifier to classify weapons with more granularity.**

clients and reducing the number of platforms it successfully deploys on. Though slow, the CLI has the advantage of not requiring both a client and server, eliminating an additional point of failure in a distributed setting. However, its sluggishness can be attributed to the extra process created per invocation and the I/O for each call, since the model file is loaded and unloaded for each process. In addition, the *Inception v3* model is approximately 200MB in size, so there is 200MB of additional I/O per parse call. Thus, using a larger model would result in more I/O time.

While we did explore JNI integrations in our interface, we didn't implement a full solution as we would have had to produce JNI glue code for all platforms and additional utilities such as Google's ProtoBuffers – a dependency required by Tensorflow – would have to be integrated with the JNI because ProtoBuffers is required for deserializing models such as ImageNet/Inception[5]. The results were encouraging both qualitatively and quantitatively; using the REST interface, we were able to index the entire 1.4 million image Memex weapons dataset in a little over four days and recommended processing post-crawl.

In addition to qualitative and quantitative evaluation of our integration techniques, we also evaluated the results of our image classification with respect to automatic processing of the Memex weapons dataset using the REST API integration. We used the *Inception v3* image classification model in our experiments [19]. This model was trained on the 2012 ImageNet dataset, which contains 1000 classes of objects such as 'tabby cat', 'lion', 'German shepherd', etc.,[8, 18] including the three labels related to our domain: 'rifle', 'assault rifle' and 'revolver'. The ten most common classes in our dataset and their frequencies are shown in Figure 3. Since the crawlers were focused on retrieving web pages and linked images related to weapons classifieds, the top classes in our dataset were found to be *revolver* and *rifle*. Automatically identifying *rifle* in this dataset was a promising result, as it gave investigators leads in discerning whether these were long guns, which have been increasingly used in weapons-related deaths over the past decade. Also promising was the second most frequently occurring class, *assault rifle*, which provides investigative leads into potentially automatic weapons in the dataset.

We evaluated the predictions using a subset of 937 images labeled by law enforcement agents, and the results are shown in Figure

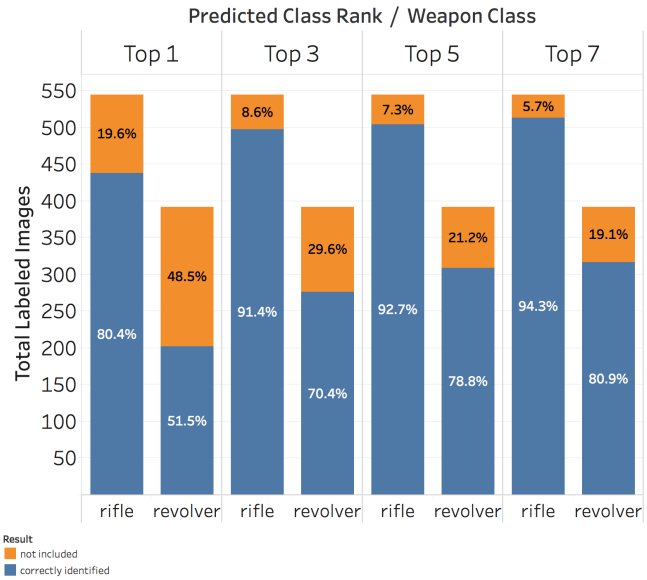


**Figure 4: Evaluation of results for the two weapon types: 'revolver' and 'rifle.' 937 images were labeled by law enforcement agents and they didn't distinguish between rifles and assault rifles. The weapon object should be the top class for all labeled images, however these results demonstrate that the model occasionally identified background objects as primary.**

4. The *top k* for $k = 1, 3, 5, 7$ considers a prediction as correct if the human-annotated label is among the set of top $k$ predicted labels. We observed that the *top 1* accuracy for the *revolver* class is considerably lower. Manual inspection of labeled images and prediction errors revealed that, due to revolvers being smaller, they are often surrounded by holders and toolkits. The *Inception* model often treated these larger surrounding objects as the predominant label rather than the *revolver* label. Hence, the error counts are reduced when $k$ is increased. Going forward, our team is interested in more granular distinctions and evaluations among weapon types. Although the *Inception v3* model identifies the assault rifle class, these labels weren't present in our evaluation data.

## 5 CONCLUSION

Our motivations for the integration of image forensics into content analysis stems from a large corpus of 1.4 million weapons related images from the DARPA Memex effort, and our goals of automatically performing image classification to identify the illegal sale of automatic weapons and other dangerous objects on the web. We integrated the widely-used Google Tensorflow toolkit, and its ImageNet/Inception v3 model with the Apache Tika framework for automated and efficient image classification and analysis. We qualitatively and quantitatively evaluated the feasibility of our integration and report on running the integration over the Memex weapons data. We also describe our process for integrating Tensorflow and Tika.

## ACKNOWLEDGMENT

This effort was supported in part by JPL, managed by the California Institute of Technology on behalf of NASA, and additionally in part by the DARPA Memex/XDATA/D3M programs and NSF award numbers ICER-1639753, PLR-1348450 and PLR-144562 funded a portion of the work.

## REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation. AAI9980887.

[3] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 363–370. DOI: http://dx.doi.org/10.3115/1219840.1219885

[4] Rob Gordon. 1998. *Essential JNI: Java Native Interface*. Prentice-Hall, Inc.

[5] Thamme Gowda. 2016. Java CPP: Deserializing tensorflow models for JNI use. https://github.com/bytedeco/javacpp-presets/issues/240. (2016). [Online; accessed 20-February-2016].

[6] gRPC. 2016. About gRPC. https://web.archive.org/web/20160828193407/http://www.grpc.io/about/. (2016). [Online; accessed 20-February-2017].

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[8] Stanford Vision Lab. 2012. Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) Synsets. http://image-net.org/challenges/LSVRC/2012/browse-synsets. (2012). Accessed: 2017-04-11.

[9] C Mattmann, G Yang, H Manjunatha, T Gowda, A Zhou, J Luo, and L McGibbney. 2016. Multimedia metadata-based forensics in human trafficking web data. *WSDM* (2016), 10–13.

[10] Chris Mattmann and Jukka Zitting. 2011. *Tika in action*. Manning Publications Co.

[11] Corianna Jacoby Mentor and Ming Chow. 2016. The Onion Router and the Darkweb. (2016).

[12] Apache Open NLP. 2016. Apache Open NLP. https://opennlp.apache.org/. (2016). [Online; accessed 20-February-2017].

[13] MIT NLP. 2016. MITIE: library and tools for information extraction. https://github.com/mit-nlp/MITIE. (2016). [Online; accessed 20-February-2017].

[14] Federal Business Opportunities. 2016. DARPA-BAA-14-21: Memex. https://web.archive.org/web/20161202033850/https://www.fbo.gov/index?s=opportunity&mode=form&id=426485bc9531aaccba1b01ea6d4316ee&tab=core&_cview=0. (2016). [Online; accessed 20-February-2017].

[15] Mark E. Phillips. 2014. Comparing Web Archives: EOT2008 and EOT2012 âĂŞ What. https://web.archive.org/web/20170220225902/http://vphill.com/journal/post/5962/. (2014). [Online; accessed 20-February-2017].

[16] Tasneem Raja. 2016. Semi-Automatic Weapons Without A Background Check Can Be Just A Click Away. (Jun 2016). http://www.npr.org/sections/alltechconsidered/2016/06/17/482483537/semi-automatic-weapons-without-a-background-check-can-be-just-a-click-away

[17] Google Research. 2016. Running your models in production with TensorFlow Servingt. https://web.archive.org/web/20170201182432/https://research.googleblog.com/2016/02/running-your-models-in-production-with.html. (2016). [Online; accessed 20-February-2017].

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. DOI: http://dx.doi.org/10.1007/s11263-015-0816-y

[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *CoRR* abs/1512.00567 (2015). http://arxiv.org/abs/1512.00567