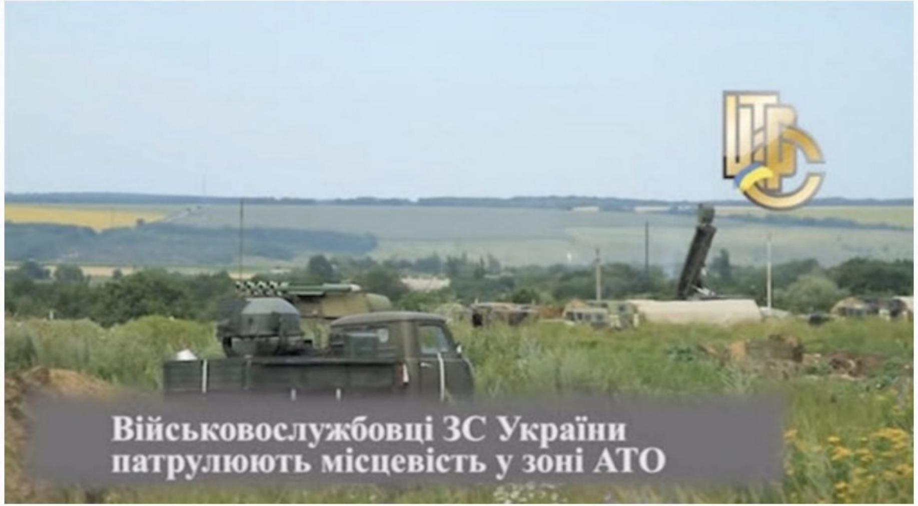
Automating Open Source Arms Tracking in Conflict Zones with Image Classification

Data Availability

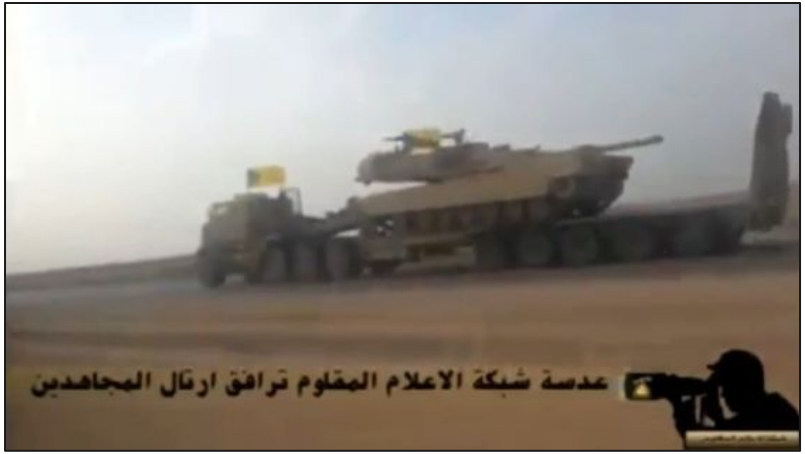
- Massive amount of open source footage and images available (YouTube, Twitter, Facebook, etc.)
- Organizations like Bellingcat have shown how powerful it is


Військовослужбовці ЗС України
патрулюють місцевість у зоні АТО

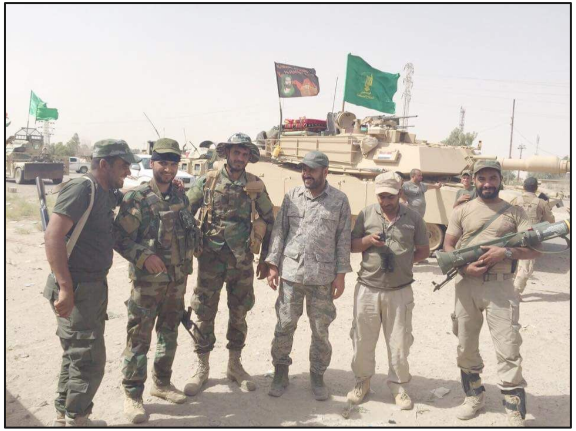
Screenshot from segment filmed by Ukrainian Military TV showing a Ukrainian Buk, aired on July 16, 2014

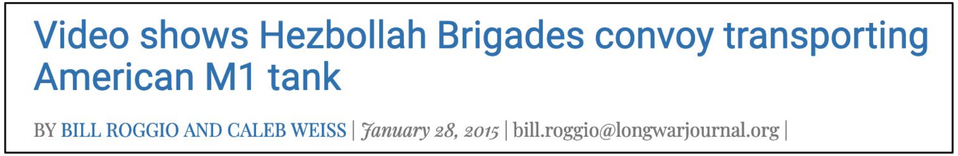
End-use Example

- Ensuring weapons or equipment legally sold by one country to another are not then sold or transferred to a third party
- One of the worst examples of this involved several M1 Abrams tanks



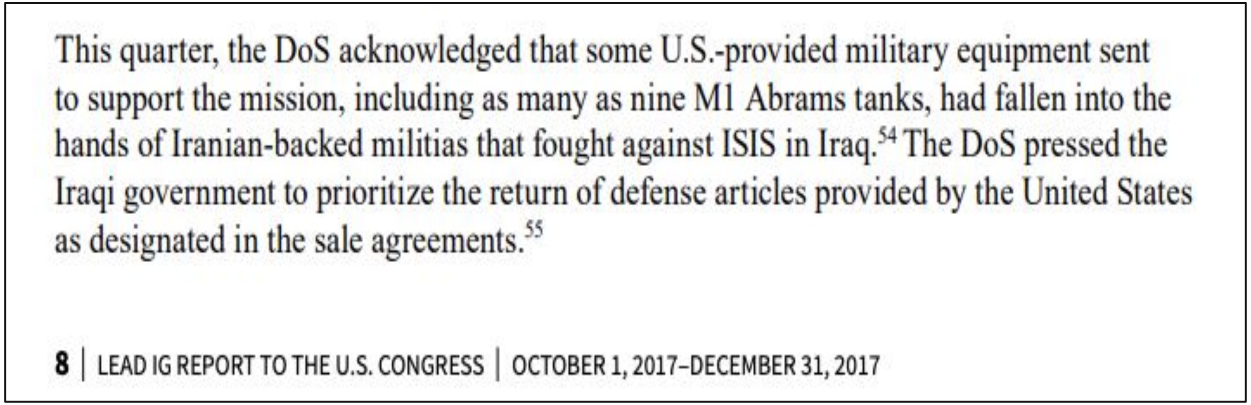
End-use Example





Video shows Hezbollah Brigades convoy transporting American M1 tank

BY BILL ROGGIO AND CALEB WEISS | *January 28, 2015* | bill.roggio@longwarjournal.org |



This quarter, the DoS acknowledged that some U.S.-provided military equipment sent to support the mission, including as many as nine M1 Abrams tanks, had fallen into the hands of Iranian-backed militias that fought against ISIS in Iraq.[54] The DoS pressed the Iraqi government to prioritize the return of defense articles provided by the United States as designated in the sale agreements.[55]

8 | LEAD IG REPORT TO THE U.S. CONGRESS | OCTOBER 1, 2017–DECEMBER 31, 2017

Problem

- One single archive of footage from the Syrian Civil War is 134 GB
- Copious amounts of data means it will take more time and manpower to sift through all of it

Solution

- Accelerate analysis by training an image classifier model on images of M1 Abrams tanks

Data

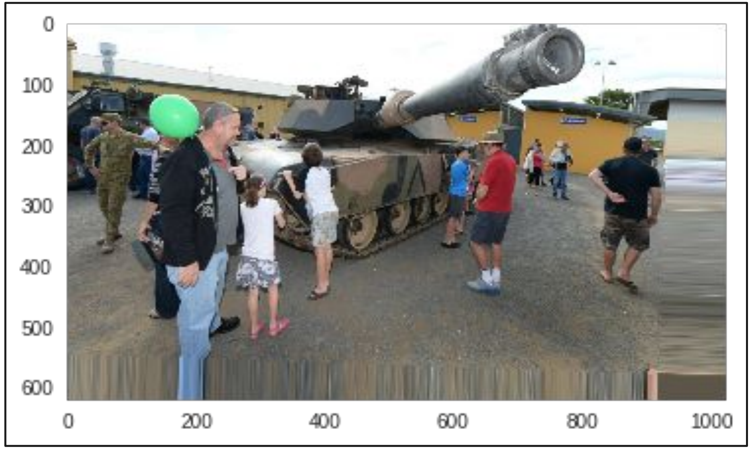
- Use the Fatkun Chrome Browser extension to download the results of a Google Image search
- Separate queries for M1 Abrams tanks and other military vehicles, especially other tanks
- Fair amount of manual cleaning after download
- ~1500 images between the two classes

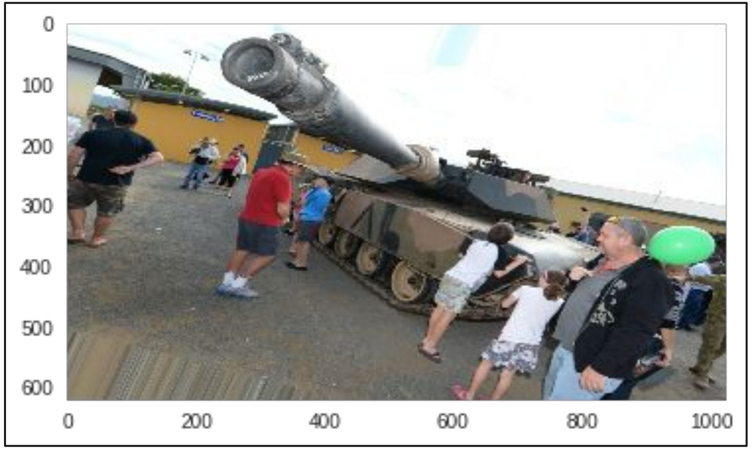
Sample Images

Sample Images

Keras’ ImageDataGenerator

- Easier for file I/O
- Will add noise and prevent overfitting
- Less computationally expensive





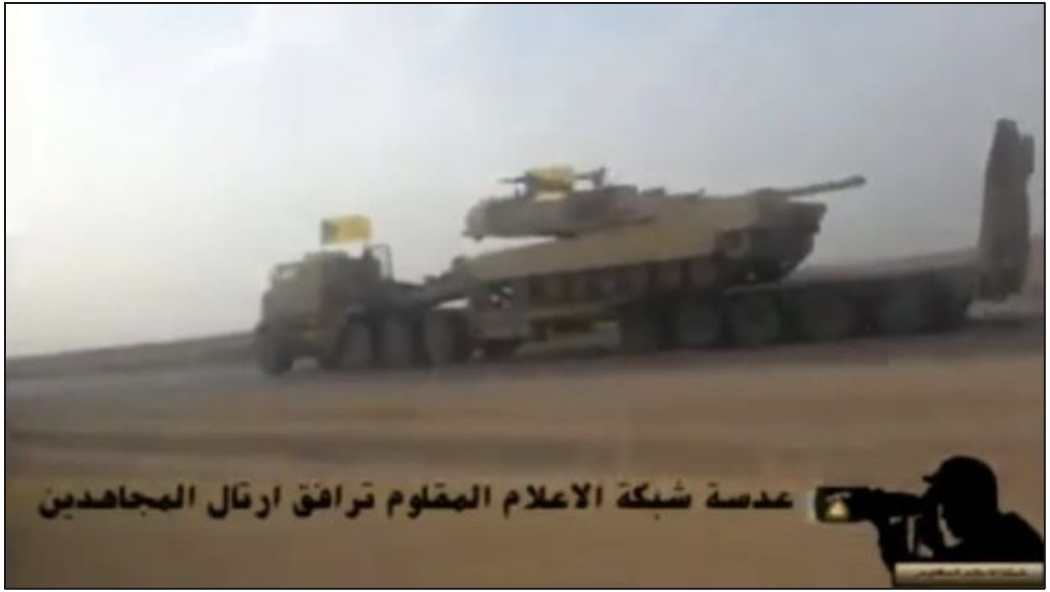
The Model

- Convolutional Neural Network to predict whether a given image contains an M1 Abrams tank or not
- Google Colab provides free GPU/TPU processing
- 4 convolutional layers
- 200 epochs
- 1.5 hours to train

Broad Performance

- Accuracy on last epoch was 0.9207 versus baseline of 0.7777 on training data
- .evaluate_generator() on test data accuracy of 0.8697 versus baseline of 0.8277
- Predicting each of the 470 test images, accuracy of 0.8617
- Type I Error: 27
- Type II Error: 38

Performance on Militia Images



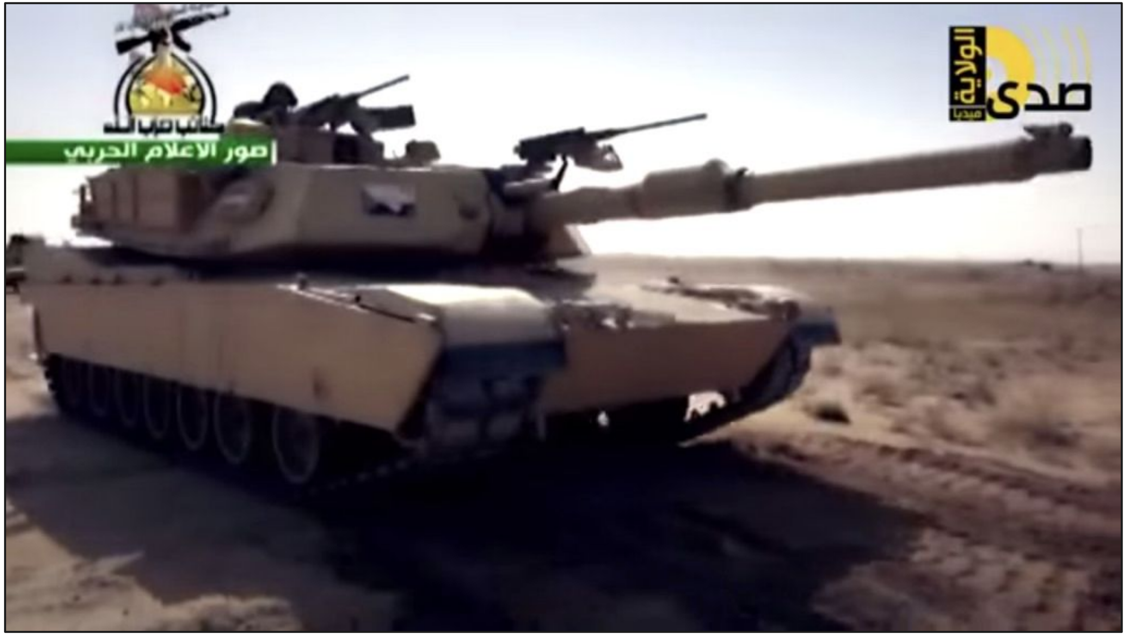
Class Prediction:

“M1 Abrams”

Prediction %:

0.9998

Performance on Militia Images



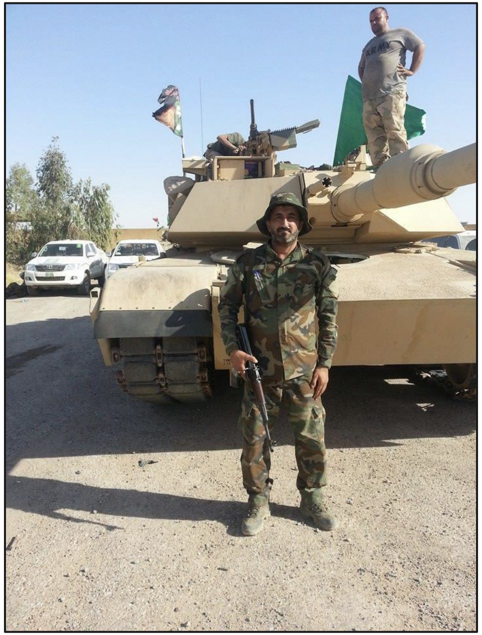
Class Prediction:

"M1 Abrams"

Prediction %:

0.9066

Performance on Militia Images



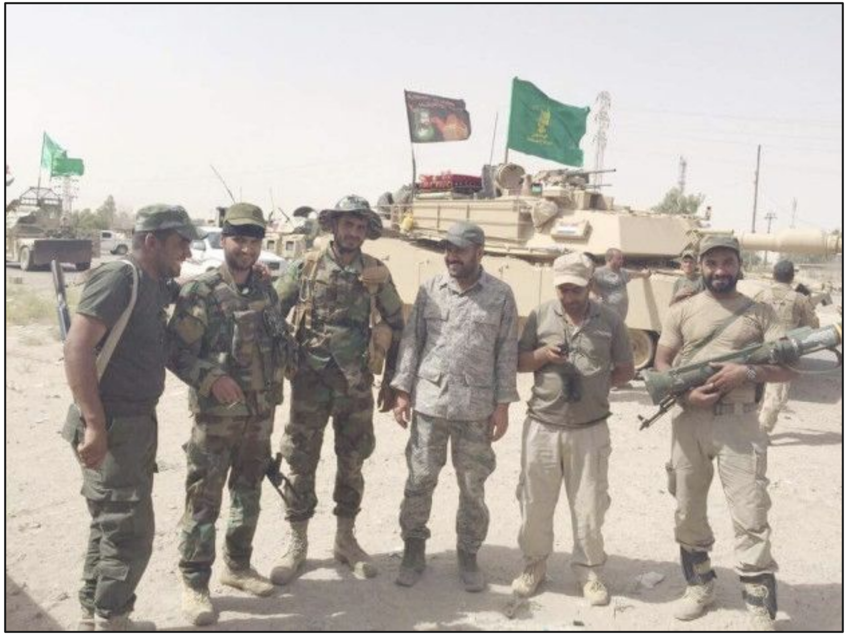
Class Prediction:

“M1 Abrams”

Prediction %:

0.9813

Performance on Militia Images



Class Prediction:

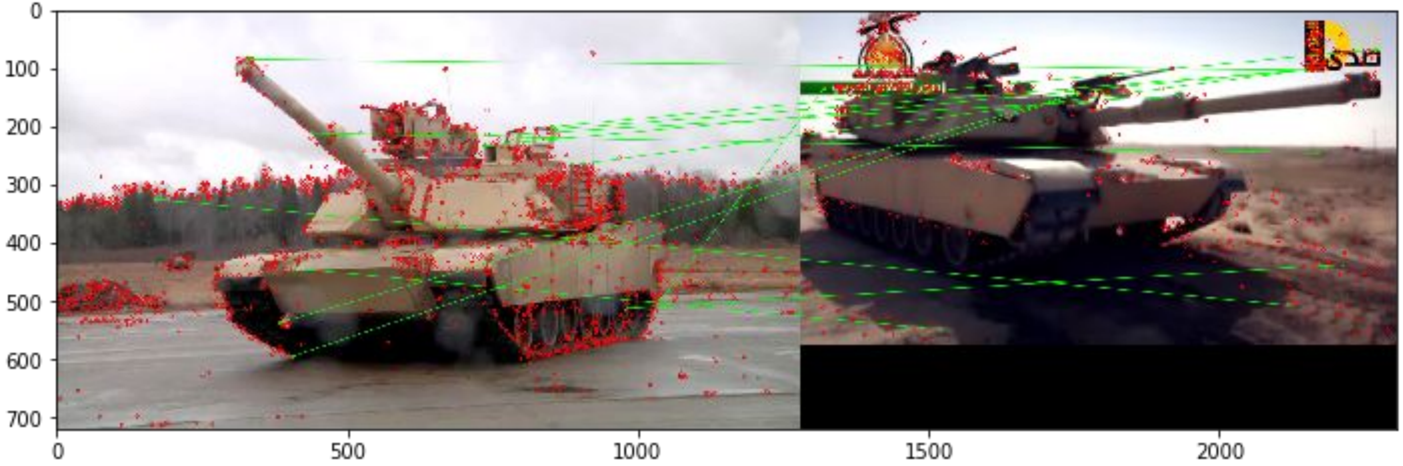
"M1 Abrams"

Prediction %:

0.6270

Feature Detection

- Scale-Invariant Feature Transform Algorithm (SIFT)
- Fast Library for Approximate Nearest Neighbors (FLANN) matcher



Conclusions

- Model beats baseline performance meaning it did learn something
- Model was able to accurately predict all four “high-risk” images
- Still missing a lot on test data

Next Steps

- Adjust model to improve accuracy and reduce false negatives
- Use YOLO or Fast/Faster R-CNN to immediately detect each M1 Abrams tank in a given image
- Incorporate other weapons systems or equipment that are of similar “high risk”
- Use Django or Flask to build a GUI so that users can input their own images or even video