Michael Davies (mld9s), Nitika Kataria (nk3rf), Mary Youssef (mry8ea)
Professor Malitsky
DS 6013
28 July 2022

NGA Draft

Abstract

The large number of social media produced images, particularly of Russian military vehicles, creates the potential for novel open-source collection and analysis of where the conflict is taking place and how it's being fought. However, manually sorting through the deluge of images is time intensive, and may be an untenable task. We implemented both an image classification, specifically ResNet50, and (multiple) object detection models, specifically YOLOv5 to explore which method better serves the goal of triaging and classifying Russian military vehicles. Ultimately, our most performant model was YOLOv5. With hyperparameter tuning and data augmentation methods, we were able to correctly classify up to 73% of images using ResNet50 and approximately 90% of images using YOLOv5. Notably, with our YOLOv5, in addition to conventional finetuning, our final model achieved good performance by ensembling two learners and implementing test-time augmentation.

Introduction

The Russia-Ukraine war has been called "the world's first TikTok war" based on the unprecedented level of user generated content, social media posts, and uploaded images (Harding). The large number of social media produced images, particularly of Russian military vehicles, creates the potential for novel open-source collection and analysis of where the conflict is taking place and how it's being fought. However, manually sorting through the deluge of images is time intensive, and may be an untenable task. Commonly used automated and timely techniques for classification of a large number of imagery and video data are convolutional neural networks (CNN's) which are typically strong performers due to pre-training on other large datasets.

In our study, we use an imagery dataset of 10 unique Russian military vehicles to train a classification as well as object detection model to classify the specific names of each vehicle using CNN models ResNet50 and YOLOv5. With hyperparameter tuning and data augmentation methods, we were able to correctly classify up to 73% of images using ResNet50 and approximately 90% of images using YOLOv5. As presumed, hyperparameter tuning as well as data augmentation improved the accuracy of our models in both the ResNet50 and YOLOv5 models. Since the YOLOV5 model performed with greater accuracy than the ResNet model, the YOLOv5 is recommended to be used to classify and detect military vehicles images.

As powerful deep learning models are becoming increasingly prevalent in the intelligence community to quickly collect open-source data for analysis, we will be using a deep

learning approach to identify Russian military vehicles. Automating a process to identify these vehicles could be used from online images scraped or video content in order to more efficiently sift through imagery intelligence. As this problem aims to classify or detect the vehicle, we researched ResNet50 and YOLOv5, discussed further below.

Background

Image Classification vs. Object Detection

Similar previous efforts to recognize Russian military vehicles from social media images leveraged the well-known ResNet50 architecture, which is an image classification deep learning model (T. Hiippala). This work showed that classification on Russian vehicles is effective, there was considerable room for improvement, possibly through changes in the architecture as well as fine-tuning. Alternatively, this effort could be done with an object detection model, which would allow for multiple objects to appear in an image.

Therefore, to compare the effectiveness of image classification versus object detection to provide value to analysts interested in tracking the conflict, we implemented both an image classification, specifically ResNet50, and a (multiple) object detection model, specifically YOLOv5. Object detection can detect the presence of multiple objects or classes in an image as well as locate the object. That is, object detection performs classification and localization. This also allows us to detect multiple objects of different classes in a single image. Also, we can point the model to video to "track" and identify multiple objects in streaming video.

Data Description

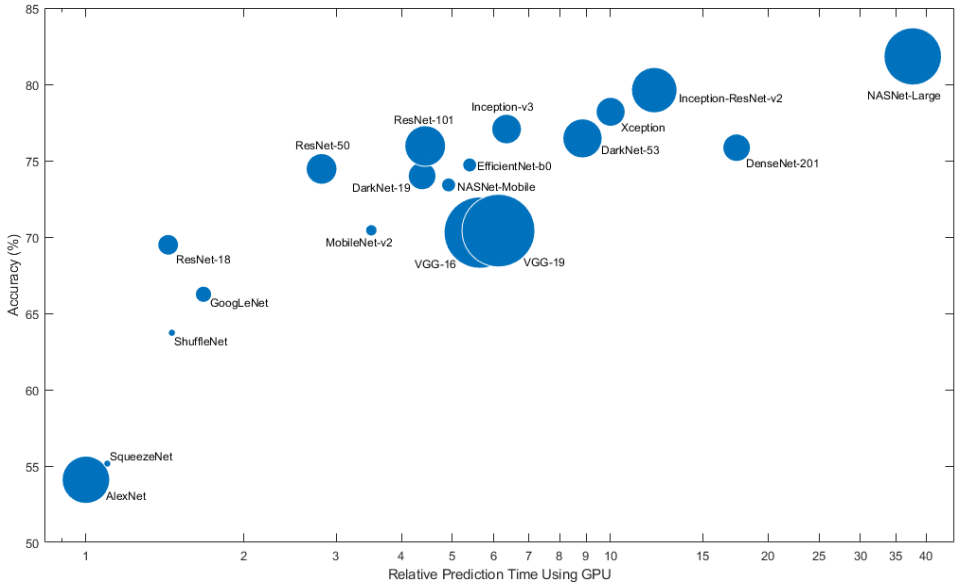
The dataset, generously provided by Tuomo Hiippala, University of Helsinki, Finland, Department of Geosciences and Geography, contains 993 images across 10 classes of Russian military vehicles. The number of images per class range from 103 to 156. Although this is not perfect class balance, it should be sufficiently close that it does not degrade the model performance. Finally, we split the data 80% training, 10% validation, 10% test. (Data for both object detection and classification is maintained here: https://github.com/MLDavies/russian-military-vehicles-annotated) For classification, we use the images as they are. However, for object detection, we need to annotate or label the images with bounding boxes around the image. We discuss annotation when detailing the YOLOv5 model below.

Model Descriptions: ResNet50 and YOLOv5

ResNet50

The image classification model we researched is ResNet50. This model won the ImageNet competition in 2015. ResNet stands for residual network and the 50 represents that the model is 50 layers deep. It is a commonly used model for image classification as well as object detection. Deep neural network models are often difficult to train due to the vanishing gradient problem which occurs when repeated multiplication makes the gradient extremely small and causes the model to stop learning. ResNet resolves this issue by skipping the connection that allows input to flow in a shortcut to an activation layer (MathWorks).

From the plot below, we see why ResNet50 is a common "go-to" model, frequently used as the baseline when comparing models (Sethi). While some models are more performant, ResNet50 achieves strong accuracy at relatively low training times.



Comparison of Models

With the implementation of ResNet50, we conducted data augmentation to the data set. Data augmentation is a technique to artificially create new training data from existing training data. Data augmentation is a powerful strategy and common in computer vision problems to upsample the size of a training dataset and inject ‘noise’ or plausible variations into the inputs. The new data is created by copying the images. Then the new images can be cropped, flipped, rotated, add grayscale, over- underexposed etc. Therefore, data augmentation helps increase the size of our dataset as well as addresses the lack of diversity in our data set. This is important because virtually all of the images in our dataset were taken on sunny days. Without diversifying our dataset, we would be training to a specific setting and the model would perform poorly on tanks taken in any other weather conditions.

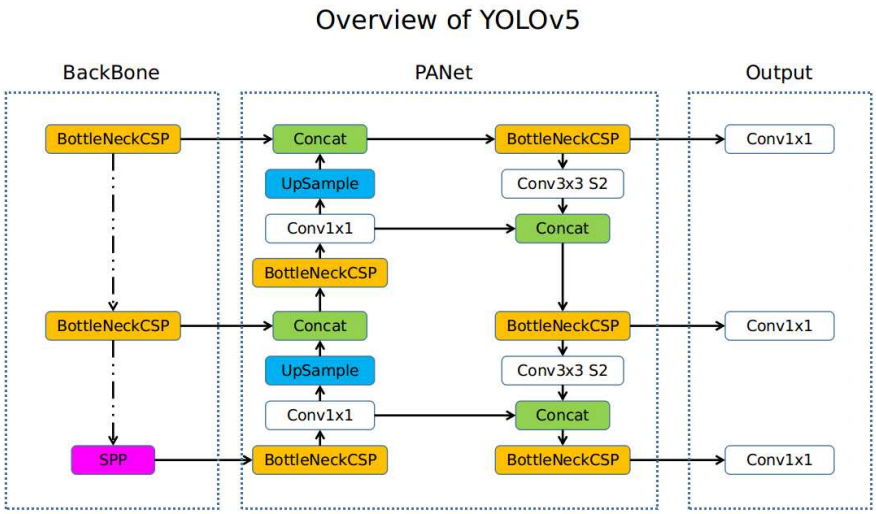
As ResNet50 models can be pre-trained using ImageNet to perform transfer learning, we will be leveraging this feature. Since our data set is not part of ImageNet, it will increase the amount of training data available for the models to allow increased accuracy. Transfer learning has become an important and popular optimization that allows rapid progress or improved performance when modeling the second task. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. Stated another way, transfer learning is an approach where pre-trained models are used as the starting point in computer vision and natural language processing tasks.

YOLOv5

At the time this project began, the state-of-the-art (SOTA) YOLO models were YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOS--each by different developers with varying goals. We chose YOLOv5, developed by Glenn Jocher of Ultralytics, because its performance is comparable to the other YOLO models, it is implemented in Pytorch, and it appeared to be under much more active development with frequently added functionality (e.g easy extraction of feature maps for model evaluation).

YOLO models are single stage object detectors that perform classification and localization. Prior work on object detection repurposes classifiers to perform detection. The original YOLO (You Only Look Once) model, a single stage detector, was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance and is extremely fast (Redmon). YOLOv5, like all single stage detectors, has three important parts:

1. **Model Backbone:** CSPDarknet53 primarily feature extraction.
2. **Model Neck:** PANet (path aggregation) primarily to generate "feature pyramids' ' and feature fusion to help the model identify the same object with different sizes and scales and - ultimately generalize to unseen data.
3. **Model Head:** The YOLO layer, the same head as YOLOv3 and YOLOv4, which generates three different output of feature maps and generates final output vectors with class probabilities, objectness scores, and bounding boxes. (M. Mayur, 2020) and (U. Nepal, H. Eslamiat, 2022)



Architecture figure on Ultralytics Github by user "seekfire".

Object Detection Challenges

Identifying Russian vehicles from images and video during combat operations presents important challenges, such as:

- **The small object problem**: Vehicles appearing in the distance or with low resolution.
    - Several data augmentation strategies have been shown to address this challenge. Of special interest to us is mosaic augmentation, invented by Glenn Jocher. Mosaic augmentation is especially useful in helping to address the small object problem. (J. Solawetz, 2020) Additionally, increasing the image size (during train and detect) is often helpful.
- **Movement and blurring**: Convoys of vehicles on the move.
    - YOLOv5 applies online blue to gray augmentations as well as "mixup", to aid in addressing these issues, to random images in the trainloader imagespace and colorspace during the augmented Mosaic.(Ultralytics documentation)
- **Occlusion**: Obfuscation created by clouds, buildings, overlapping vehicles, blast debris, smoke, damaged or destroyed vehicles and vehicles partially buried in dirt.
    - Again, as was shown in the study on UAV Detection and Classification, which used infrared videos with complex backgrounds such as buildings, clouds, and trees, data augmentation such as scaling, color space adjustments, and mosaic augmentation aided in addressing this challenge.(F. Dadboud et al, 2021)

Data Annotation

Object detection requires that the training images be annotated or "labeled" with bounding boxes around each class of vehicle. For this, several open source software packages are available. We chose to use the Roboflow platform, an online service that allows a team to divide images and compile when complete. Each object detection requires a specific file type

and format for labels. YOLOv5 requires labels in a .txt file type in a particular format. In total, we have 1254 annotated classes across the 993 images. (See an example below) Note, by annotating the final hold-out images, we can create a confusion matrix to evaluate the performance on unseen data.

Methodology/Implementation

ResNet50

For our project we implemented a few machine learning techniques when building the image classification ResNet50 models. Before building a customized model with pre-trained weights, first a base model was built with only the pre-trained weights as a baseline for the customized ResNet50 models. For these models, the data is split with 80% for training, 10% for validation, and 10% for testing. We chose to use a train-validation-test split rather than K-fold cross validation because the computational cost associated with K-fold cross validation is too great. Image data augmentation was applied to the images by rescaling so that the pixel values can be normalized to [0,1]. They were also randomly flipped horizontally and had the shearing transformation applied on randomly. A consistent batch size of 16 was set and the images were set with the width and height of 224 for the models.

The base model is implemented with the pre-trained weights of ImageNet. Along with the 50 layers that exist within the ResNet50 architecture, a hidden layer consisting of the flatten layer is added. The flatten layer is added to convert the multi-dimensional arrays from the ResNet50 architecture into a flattened one-dimensional array (McLean). As an output layer, a dense layer is added that has an activation function of softmax. The last layer in a classification model commonly has the softmax function to transform an unconstrained n-dimensional vector

into a valid probability distribution (NodePit). The model is then compiled using an optimizer, loss function, and the metric to be measured. For the base ResNet50 models, the optimizers *SGD* and *RMSprop* were used. The loss function used is *sparse_categorical_crossentropy* for all the models since the image labels are provided as integers (TensorFlow). The metric that was used is *accuracy* for all the models. To help with the optimization of the model, a function called LRFinder is used to find the ideal learning rate for the specified model (Pai). This LRFinder works by monitoring the loss function as the learning rate is exponentially increasing within the set range of the starting and ending learning rate. The learning rate that is found at the lowest loss function value is set as the model's learning rate through the Keras backend. The best learning rate is applied when the model is set to compile (Pavel).

Once the models have been compiled, they are set to run for 100 epochs. During these 100 epochs, the model is trained on the training data and validated using the validation data. Once the model has been trained, two plots are displayed, one showing the accuracy values of the training and validation set and the other showing the loss values of the training and validation set during the length of the total number of epochs. The test set is used to output the confusion matrix and is evaluated to display the accuracy, precision, recall, and f1-score of the model.

The customized ResNet50 models consist of the output of the base model and includes additional layers in an attempt to improve the performance of the model. Additional layers consist of BatchNormalization layers, Dropout layers, additional Dense layers with varied number nodes to analyze the effect of the layers on the model's performance. With these customized models, there's also the ability to input the number of layers to unfreeze from the base model if desired to. Like the base model, the customized models are compiled using the *RMSprop* optimizer and run through the LRFinder to use the best learning rate. Once the model is fit and completes the training with 100 epochs, the plots, confusion matrix, and evaluation metrics are displayed as stated earlier.

YOLOv5

During fine-tuning, we conducted numerous experiments with changes in hyperparameters, anchor boxes, model size as well implementing model ensembling and test-time augmentation (TTA), ultimately developing a highly performant model. Interestingly, given that military vehicles are not represented in the COCO dataset, we did not expect transfer learning to improve performance. However, all tests show some improvement when we implemented the pretrained model. Ultimately, the models plateaued between 100 and 150 epochs, so we trained all models at 150 epochs. Also, as mentioned, we implemented a few novel strategies:

- **Model Ensembling**: Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. (Vijay Kotu, Bala Deshpande, 2015) I tested the performance of two models, YOLOv5s and YOLOv5l. The latter was much more performant. However I found by ensembling the two models, and testing on the final hold-out dataset, we can achieve even better performance by ensembling the two models during inference. (See the

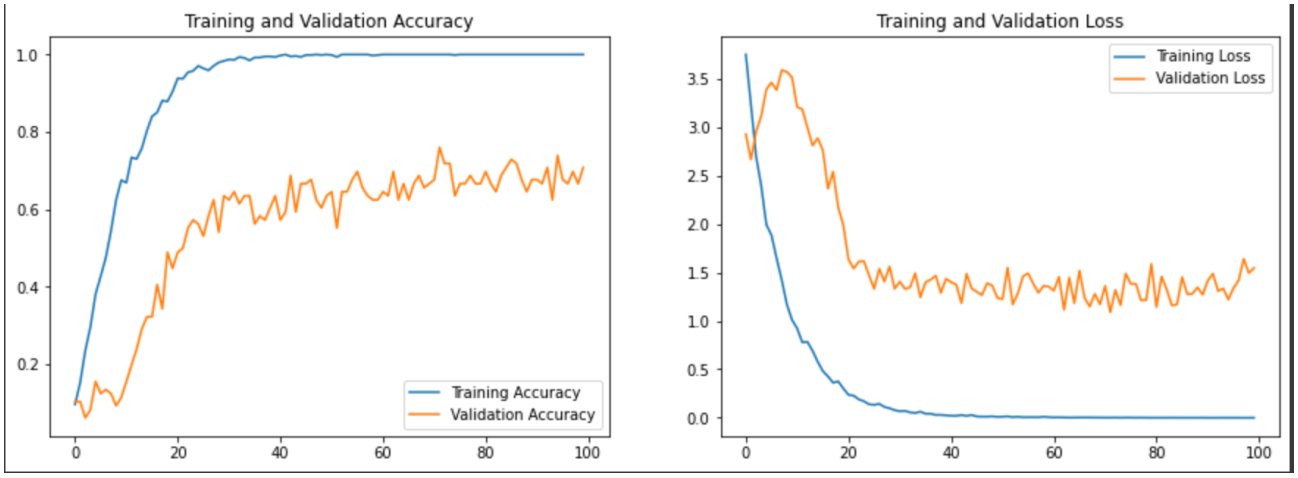
confusion matrices during validation below) The primary difference between the YOLOv5s and YOLOv5l is that the former has 7.2 million parameters whereas the latter has 76.8 million parameters.

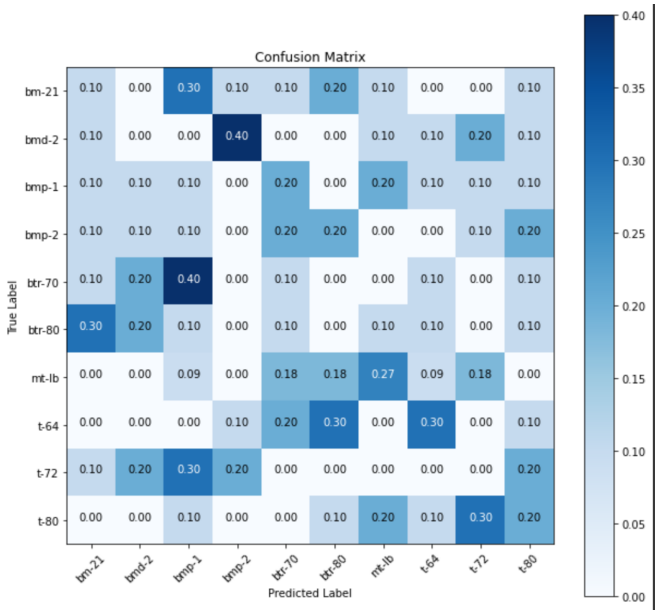
- **Test-Time Augmentation (TTA)**: During final inference, we can further boost generalization by applying test-time augmentations (TTA). (J. Nalepa, M. Myller, M. Kawulok, 2019) TTA is the process by which each test image is augmented (horizontal flip and 3 different resolutions), and then fed into the trained model. As such, TTA simulates re-fitting the entire pipeline (including the model) once new data is available. (H2O.ai Driverless AI) The final prediction during inference is an ensemble (average) of all the augmented images.

Results

ResNet50

The base ResNet50 model was run using both SGD and RMSprop optimizers. From these two models, RMSprop performed significantly better with accuracy of 64% while the model with the SGD had an accuracy of 37%, therefore RMSprop was used for the customized ResNet50 models. The resulting plots, confusion matrix, and metrics can be seen below for the base ResNet50 model using the RMSprop optimizer. The Training and Validation Accuracy plot shows the model overfitting and not generalizing the data since the training accuracy is significantly greater than the validation accuracy.



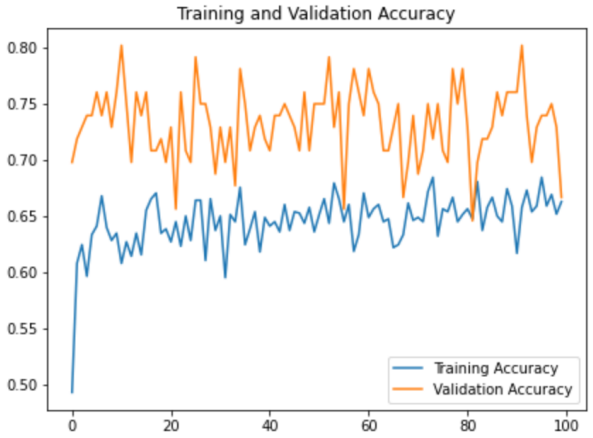


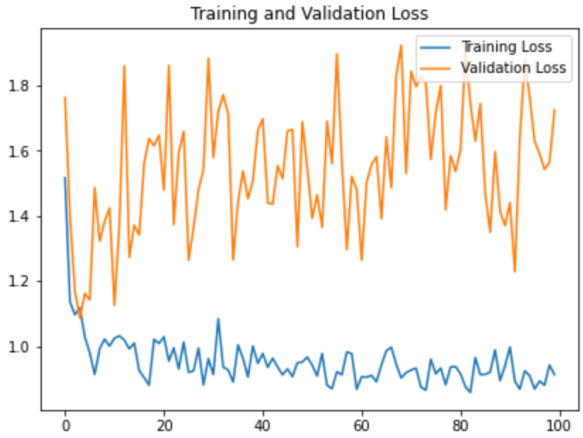
	precision	recall	f1-score
0	0.11	0.10	0.11
1	0.00	0.00	0.00
2	0.07	0.10	0.08
3	0.00	0.00	0.00
4	0.09	0.10	0.10
5	0.00	0.00	0.00
6	0.30	0.27	0.29
7	0.33	0.30	0.32
8	0.00	0.00	0.00
9	0.17	0.20	0.18

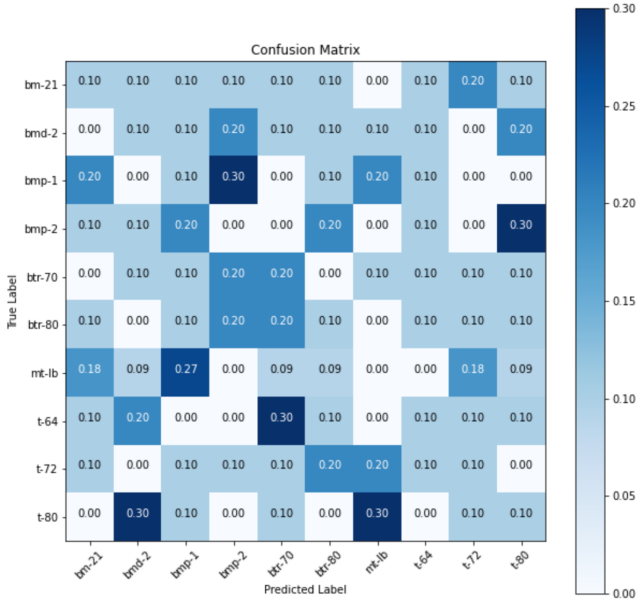
After experimenting with the types of layers and number nodes, the model with BatchNormalization, Dropout, and Dense layers had the highest accuracy of 73% when evaluated on the test data set. This model also had its last 4 layers of the base model unfrozen to fine-tune the model. The architecture of the model can be seen in the image below.

```
x = model_resnet.output
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(64, activation ='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(32, activation ='relu')(x)
x = BatchNormalization()(x)
output_layer = Dense(n_classes, activation='softmax')(x)
```

The resulting plots, confusion matrix, and metrics for this model can be seen below in the following figures. The Training and Validation Accuracy plot shows the validation accuracy being greater than the training accuracy which may be due to the fact that Dropout layers are being used to set 50% of the features to zero.







	precision	recall	f1-score
0	0.11	0.10	0.11
1	0.10	0.10	0.10
2	0.08	0.10	0.09
3	0.00	0.00	0.00
4	0.17	0.20	0.18
5	0.10	0.10	0.10
6	0.00	0.00	0.00
7	0.12	0.10	0.11
8	0.11	0.10	0.11
9	0.09	0.10	0.10

YOLOv5

mAP

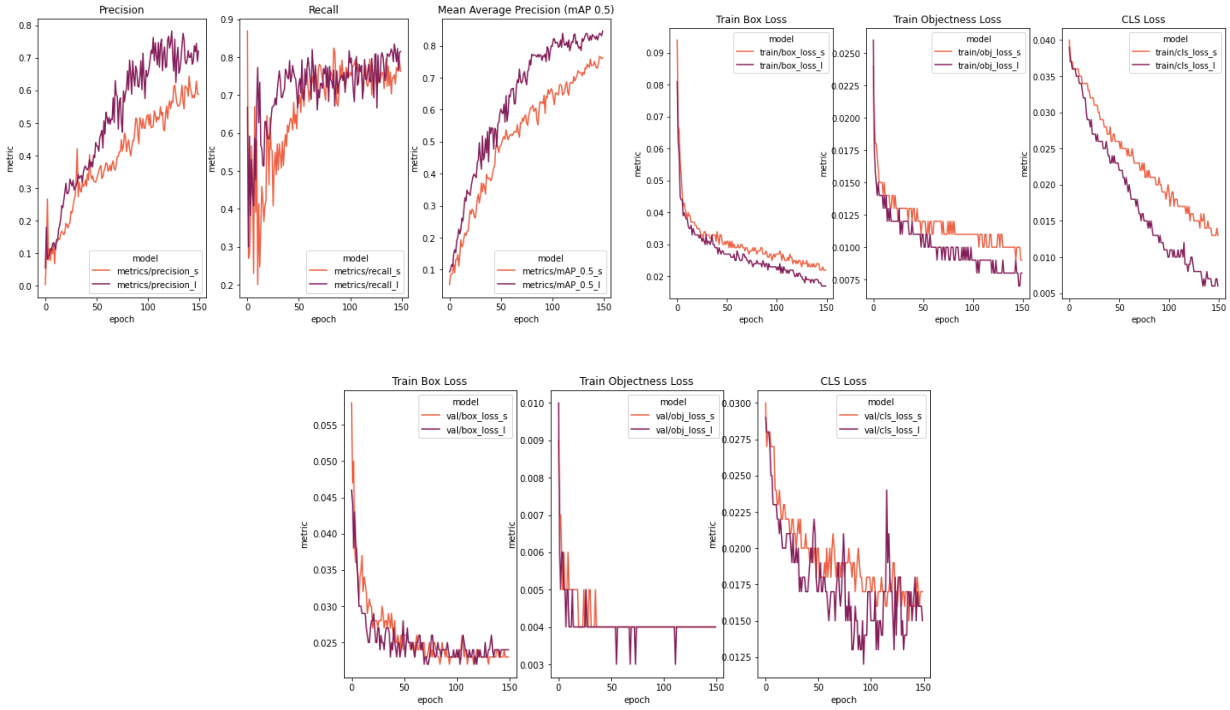
The computer vision community has converged on the Mean Average Precision ("mAP") metric to compare the performance of object detection systems. Object detection models make predictions in terms of a bounding box and a class label. First, the Intersection over Union (IoU) measures the amount of predicted bounding box that overlaps with the ground truth bounding box divided by the total area of both bounding boxes. Next, it calculates the average precision (AP) for each class individually across all of the IoU thresholds. Then the metric averages the mAP for all classes to arrive at the final estimate. (J. Solawetz, 2020)

Additional important metrics, which we don't have the space to cover here, include: box loss, objectness loss and classification loss.

Assessment of our model

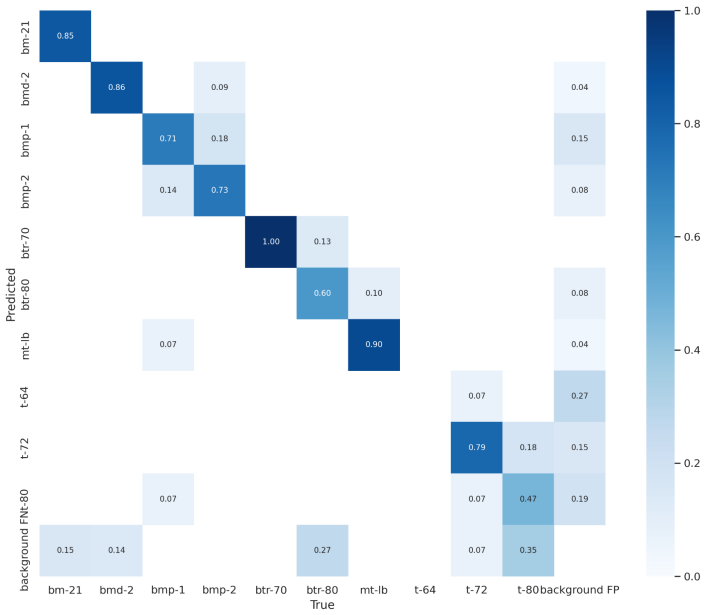
The model improved swiftly in terms of precision, recall and mAP before beginning to plateau after about 100 epochs. The mAP for the YOLOv5 (large model) was approximately 85 percent at 150 epochs. The box, objectness and classification losses of the validation data also showed a rapid decline until around epoch 100. Importantly, the small model only achieved approximately 75 percent mAP. We shall see later that assembling these models significantly improves the performance.

Evaluation/Performance Metrics

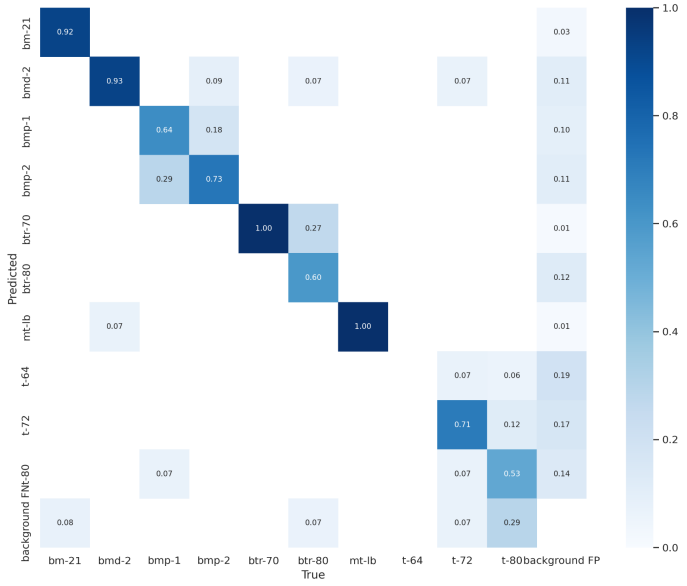


Validate: Testing on Unseen Data (Final Hold-out Data)

Now we predict our final hold-out data and check for accuracy. Here we just predict with our large model and ensembled model, both with TTA. As expected, there is a slight decrease in performance on unseen data when using the large model alone (relative to the training results). However, ensembling the two models resulted in a highly performant model, which can be seen in the confusion matrices below. However, model accuracy was not consistent across vehicle types. The bm-21, bmd-2, btr-70, and mt-lb fell between 92 and 100 percent. However, the main battle tanks (t-64, t-80) and bmp-2 were harder to accurately classify.



Confusion matrix for large model



Confusion matrix for ensembled model

Using the final model to locate and classify (on images and video)

We can simulate deployment by predicting on the final hold-out set (with TTA and ensembling). The images below show a sample of predicted bounding boxes and classification at a given probability.

Detect on Video Data with URL

We also pointed our ensembled model to a URL of an online video depicting a convoy of Russian military vehicles. The resulting predictions can be found at this URL: https://youtu.be/TW6sQBz0dPs. An image from the video is shown here:

Discussion

As we've shown, the ResNet50 performed with an accuracy of 73% after experimenting with the model's architecture. The experimentation allowed us to see the effects of the type of layers and number nodes on the performance of the models. Furthermore, the fine tuning method of unfreezing some layers showed how the performance can be enhanced with this small change. In terms of our YOLOv5, unsurprisingly the large model performed better than the small model. However, we were also able to develop an impressive multiple object detection model by ensembling the two models. Additionally, we employed test-time augmentation, resulting in a highly performant model that can be deployed on images and video.

Conclusion

Developing methods to automate and quickly sort through open source images to prioritize relevant intelligence has become increasingly important for countless agencies. We

have implemented two CNNs, ResNet50 and YOLOv5 for classification and object detection, respectively, to sift through and tag Russian military vehicles. The YOLOv5 model performed slightly better than the ResNet50 model, despite challenges given object detection such as movement and blurring. Building open the models and including more data could allow for better accuracy, and potentially tagging and capturing more detail within the vehicle, such as insignia.

References

Harding, Luke. “The First Tiktok War: How Are Influencers in Russia and Ukraine Responding?” The Guardian, Guardian News and Media, 27 Feb. 2022, https://www.theguardian.com/media/2022/feb/26/social-media-influencers-russia-ukraine-tiktok-instagram?utm_source=pocket_mylist.

Hiippala, Tuomo. “Recognizing Military Vehicles in Social Media Images Using Deep Learning.” 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), 2017, https://doi.org/10.1109/isi.2017.8004875.

MathWorks. “Pretrained Deep Neural Networks.” Pretrained Deep Neural Networks - MATLAB & Simulink, https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html.

McLean, Kevin. “How to Use Keras.layers.flatten().” Medium, Medium, 29 Aug. 2021, https://kevinmlean.medium.com/how-to-use-keras-layers-flatten-c3f29ed1b686.

Nelson, Joseph. “Your Comprehensive Guide to the Yolo Family of Models.” Roboflow Blog, Roboflow Blog, 19 July 2022, https://blog.roboflow.com/guide-to-yolo-models/.

NodePit. “Keras Softmax Layer.” NodePit, 11 June 2022, https://nodepit.com/node/org.knime.dl.keras.base.nodes.layers.advancedactivation.softmax.DLKerasSoftmaxLayerNodeFactory.

Nepal, Upesh, and Hossein Eslamiat. “Comparing Yolov3, Yolov4 and Yolov5 for Autonomous Landing Spot Detection in Faulty Uavs.” Sensors, vol. 22, no. 2, 2022, p. 464., https://doi.org/10.3390/s22020464.

Pai, Nanda Kishor M. “Improving Model Accuracy with Transfer Learning, Data Augmentation, LR Finder, and Much More.” Paperspace Blog, Paperspace Blog, 12 Oct. 2021, https://blog.paperspace.com/improving-model-accuracy/#lr-finder-finding-the-perfect-learning-rate.

Redmon, Joseph, et al. “You Only Look Once: Unified, Real-Time Object Detection.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, https://doi.org/10.1109/cvpr.2016.91.

Rodriguez, Emanuel. “Resnet50 Image Classification in Python.” A Name Not Yet Taken AB, 28 May 2020, https://www.annytab.com/resnet50-image-classification-in-python/.

Sethi, Shilpa, et al. “Face Mask Detection Using Deep Learning: An Approach to Reduce Risk of Coronavirus Spread.” Journal of Biomedical Informatics, vol. 120, 12 Aug. 2021, p. 103848., https://doi.org/10.1016/j.jbi.2021.103848.

Surmenok, Pavel. “Keras_lr_finder/lr_.” GitHub, 4 July 2020, https://github.com/surmenok/keras_lr_finder/blob/master/keras_lr_finder/lr_finder.py.

Team, Towards AI. “Yolo v5 - Explained and Demystified.” Towards AI, 1 July 2020, https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8A-%E2%80%8Aexplained-and-demystified.

TensorFlow. “Tf.keras.losses.SparseCategoricalCrossentropy .” TensorFlow, 19 July 2022, https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy.

“YOLOv5 Documentation.” Ultralytics, https://docs.ultralytics.com/.