TL;DR:
The files used to create the training set are: "create_dataset.py", and "box_training_imgs.py". The training image files are located at "train_data", annotated image files are located at "annotations". The plot of Training and Testing mAP vs epochs is located at "imgs/map_epoch.png". The video of the object detector in action is located at "videos/yolov2_detector.mp4". The object detector source code is located in darkflow-master, and the remaining supplemental/dependency files are: "darkflow-master/bin/yolo_weights/tiny-yolo-voc.weights", "darkflow-master/cfg/tiny-yolo-voc-1c.cfg", "gen_xml.py", "rename.py", "darkflow-master/classify.py," "darkflow-master/plot_map.py"

Installation:
This repository uses Python 3.5
git clone this repo
pip install -r dependencies.txt

Create Dataset:
The first step in this software pipeline is to download a bunch of random images to serve as the background images and then a series of time magazine cover images to randomly overlay onto the background images. This could be done with a script to scrape images but I decided to use the Fatkun Batch Download Image Chrome extension (https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahlikiabnchcpehcpkdeckfgnohf?hl=en) to focus more time on the object detector. The Time magazine covers that I downloaded came from the following google searches: "Time Magazine Cover", "Time Magazine", "Old Time Magazine", and "History Time Magazine Covers." The background images came from the following Google searches: "Cool Cars", "Outdoor Activities", and "Images."

Next, I created the training data set by overlaying the time magazine cover images to a random position on every background image using create_dataset.py. Example usage: python3 create_dataset.py <background image folder path> <overlay image folder path>

In order to build an object detector that recognizes time magazine covers, I created “box_training_imgs.py” Usage: python3 box_training_imgs.py, this module allows you to draw a box around each Time magazine cover in each image of the data set. Once the box's are drawn, the user must press the 'n' key to receive the next image. After the location of each Time cover is labeled, the position of the magazine is saved in an xml file in the "annotations" folder. Files in this annotation folder are later fed to the object detector during training.

Train YOLOv2 on Time Magazine Covers:
The assessment suggested to use Keras or Cafe with darknet but I like Tensorflow so I chose to use darkflow (https://github.com/thtrieu/darkflow)

Config and Weights:
I chose to use the Tiny YOLOv2 VOC weights and config files. Tiny YOLOv2 is less accurate but much faster than YOLOv2 because I don't have the recommended GPU requirements for the full YOLOv2. Tiny YOLOv2 VOC is trained on the Pascal VOC data set which is a standard detection set that uses 20 image classes. According to the YOLO9000 paper (https://arxiv.org/pdf/1612.08242.pdf), in comparison to other data sets like COCO, Pascal VOC achieves a higher mAP value.

Training:
Before training, I changed the darkflow-master/labels.txt file to have only the "time_cover" class label, I resized the number of filters in the last convolutional layer to be 30, which is just num * (number of classes + 5), and I changed the

number of classes in the region layer to 1, because we are only looking for Time covers in each image. For training, I fed the detector 350 of the Time magazine annotated image set with gpu processing and 300 epochs. In the end, I noticed that the detector stopped improving between steps 1250 to 1375 (epochs 78 to 86) with a loss of 0.3 so I stopped training at epoch 86. You can train the detector by executing:

```
"cd darkflow-master"
"pip install -e ."
"flow --model cfg/tiny-yolo-voc-1c.cfg --load bin/yolo_weights/tiny-yolo-voc.weights --train --annotation ../annotations/ --dataset ../train_data --gpu 1.0 --epoch 300"
```

Load YOLOv2 to classify new data:

To demonstrate the detector, I made the video, "videos/yolov2_detector.mp4". This video shows how to execute darkflow-master/classify.py which takes novel image data and displays YOLO's prediction and bounding box for any and all Time covers that appear in the photos. An example usage for classify.py: python3 classify.py <validation image directory>

Plotting Training and Testing mAP vs Epochs:

The plot of Testing mAP and Training mAP doesn't start to differ until after the 30th epoch when the training data shows a slowly growing increase in mean average precision (mAP) as the epochs increase. This is due to the object detector overfitting on the training dataset, the best measure of mAP performance is the Testing mAP which reaches 90.64% accuracy at epoch 86! In the future, we can improve this accuracy by training on more images (I used 350 training, and 200 validation) and also spending more time generating precise annotation label files that have a tighter boundinx box around each image.
Note: see weight file explanation above.

Supplemental Files:

box_training_imgs.py, create_dataset.py, gen_xml.py, rename.py, darkflow-master/classify.py, darkflow-master/plot_map.py