CS640 Project 2 Report

Topic 2: News Image Classification

Rongyu Wang
Xiankang Wu
Maya Shen

I . Problem Definition

Nowadays journalists are not the first ones to record an emergency event but any civilians with phones, and they send to social media almost immediately when such event takes place in their proximity. Such examples can be found as Twitter posts during the Paris 2015 Bataclan event, 2013 Boston Marathon Bombing etc.; the use of hashtags make these instances searchable.

Here, emergency event is defined as a sudden, urgent, usually unexpected incident or occurrence that requires an immediate reaction or assistance for emergency situations faced by the recipients of public assistance in order to bring the situation under control and to restore normality. [1]

For this project, we would like to build a system that looks for such event in a pool of images originated from social media feeds. The system consists of two parts: first a deep neural network trained for object detection, secondly, an expert system that yields an emergency level score[2] based on the object or event indicated by object(s) detected from images.

To reach our goal, we decided to try three different approaches for building the deep neural network, two using transfer learning techniques and one from reproducing the work from the original paper which Topic 2 is based on.[3] We aim at comparing the approaches, as well as gaining a deeper level of understanding on Convolutional Neural Network structure that's heavily used in solving Computer Vision problems.

II. Background Survey

Understanding issues such protest and violence had been a critical topic within research field crossing different academic disciplines. For example, merging with political science, there are great efforts attempted to tackle these problem through lens of social media using computer vision techniques: Liu[4] applied facial attribute classification to understanding appeals from protesters; other researchers[5] tried to analyze photographs and portraits of particular politicians. Public opinions and their social media presence are also being examined, for example:

---

[1] Emergency Event Law and Legal Definition - https://definitions.uslegal.com/e/emergency-event/

[2] Disclaimer: for the emergency level score, we do not wish to build a system that can define whether police should react or not, since we're not expert in criminal activities etc., but from a civilian's perspective, if such things happen around us, do we feel fear or panick, and do we wish law enforcement to respond in a timely manner.

[3] Protest Activity Detection and Perceived Violence Estimation from Social Media Images - https://arxiv.org/abs/1709.06204

[4] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning.

[5] Quanzeng You, Liangliang Cao, Yang Cong, Xianchao Zhang, and Jiebo Luo. 2015. A multifaceted approach to social multimedia-based prediction of elections. IEEE. Transactions on Multimedia 17, 12 (2015), 2271–2280.

processing emotion, global mobilization, immigration[6] etc.. Thanks to these work in progressing human perception of media content as well as taking more advantages of technology advancement, we're able to build more intelligent systems.

We used Transfer Learning heavily in our approaches. Here's a good definition we found describing Transfer Learning and situation to apply "Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task."[7] It's particularly useful in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given training a deep neural network requires large amount of training data, is computationally expensive and requires long period of time to find the optimum weight even on GPUs.

There're a few approaches to adapting Transfer Learning; first the pre-trained network can be used as a fixed feature extractor, this is accomplished by removing the last fully connected layer, supplement with our own desired outputs and treat the rest of the network as a fixed feature extractor for the new dataset. Secondly, there's the fine tuning approach where the weights of the pretrained network are fine-tuned by continuing the backpropagation process.

Fine-tuning is a challenging process, and has some techniques involved in it, for example, there's the choice to fine-tune all layers of the network, or keeping some layers fixed and only fine-tune the higher level portion of the network. The second way can mitigate overfitting concerns given earlier layers of a deep neural network contain more generic features that are useful in computer vision tasks in general, and later layers are more specific to task on hand. The size of the new dataset as well as its similarity to the original dataset the network it's trained on also provide crucial roles in determining an efficient fine-tuning technique.

Please see more work we did during background investigation in file Brainstorming.pdf.

III. Baseline Reproduction

We've reproduced 3 baseline approaches:

- Reproducing baseline solution from paper *'Protest Activity Detection and Perceived Violence Estimation from Social Media Images'* by *Donghyeon Won, Zachary C. Steinert-Threlkeld, Jungseock Joo*
- Transfer Learning for Multi-labelled Classification
- Transfer Learning for Single Class

[6] Lefteris Anastasopoulos and Jake Williams. 2016. Identifying violent protest activity with scalable machine learning *. (2016). http://scholar.harvard.edu/ janastas

[7] Machine Learning Mastery by Jason Brownlee: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Following are the breakdowns of the 3 approaches respectively.

III-1. Reproducing Baseline Solution from Paper

- **Goal:** use Convolutional Neural Network to predict whether there is a protest, violence rate, and other visual attributes of a given news picture.
- **Method:** Use GPU to train a CNN model based on the UCLA_Protest_Image training set, both picture and annotation.
- **Evaluation:** calculate the average square error of violence as accuracy. Then train a classifier annotating Emergency level of a news image by violence rate.

We choose UCLA_PROTEST_IMAGE_DATASET as training set since the annotation for UCLA dataset is more well rounded containing useful labels such as Group_20 and Group_100 that helps solve our defined problem. With this dataset and baseline approach, we can detect emotional sentiment thus improving prediction accuracy for violence and protest estimation. The UCLA dataset contains 40,764 images, among which 11,659 images are protest images.

Here is how we define the annotations:

- Protest: Whether there is a current protest in the image or not.
- Violence: How violent the image shows. It is more an estimation and evaluation than exact numbers.
- Sign: Whether a protester is holding a visual sign or not(maybe on paper, panel or wood).
- Photo: Whether a protester is holding a photograph or an individual or not.
- Fire: Whether there is smoke or fire in the scene.
- Police: Whether there are policemen or troop in the scene.
- Children: Whether there are children involved in the scene.
- Flag: Whether there are flags in the scene.
- Night: Whether it is during the night-time or in the daytime.
- Shouting: Whether there are people shouting in the scene.

According to Won's paper[8], we choose our model based on a 50-layer ResNet, which consists of 50 convolutional layers with batch normalization and ReLU layers. The features computed through convolutional layers are shared by linear layers for multiple classification(i.e., protest, violence, sign, photo, fire). After setting training epoch up to 20, we can get prediction accuracy at 92%.

[8] Donghyeon Won, Zachary C. Steinert-Threlkeld, Jungseock Joo. 2017. Protest Activity Detection and Perceived Violence Estimation from Social Media Images. In Proceedings of the 25th ACM International Conference on Multimedia 2017.

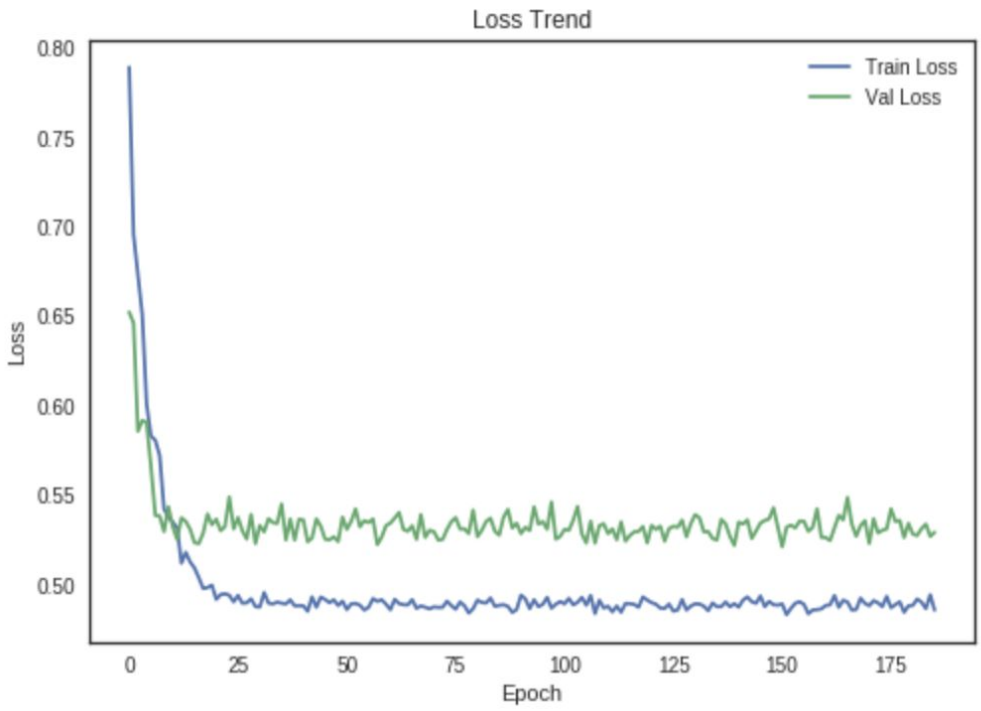
Layer	Output size	Building blocks
conv1	112 x 112	7 x 7, 64, stride 2
conv2	56 x 56	3 x 3 max pool, stride 2
		|1 x 1, 64 | |3 x 3, 64 | x 3 |1 x 1, 256|
conv3	28 x 28	|1 x 1, 128 | |3 x 3, 128 | x 4 |1 x 1, 512 |
conv4	14 x 14	|1 x 1, 256 | |3 x 3, 256 | x 6 |1 x 1, 1024 |
conv5	7 x 7	|1 x 1, 512 | |3 x 3, 512 | x 3 |1 x 1, 2048 |
pooling	2048	average pooling
classification	13	[protest, violence, visual attribute(10)]

Above is the architecture of our model. As final classification, we get a row of label prediction, respectively: [protest, violence, visual attribute(10)].

- **Steps:**
  - **Step 1:**
    Train the Convolutional Neural Network based on training set. Randomly produce weight for each convolutional layer, then set training epoch = 200, learning rate = 0.002, and batch size = 32. Run the training process on GPU on platform of CUDA and deep learning packages like Pytorch, Torchvision, and OpenBLAS.

    Each iteration completes, the train.py will use torch.save() function to keep record of every model separately. When the model performs better than previous one, it will be saved as the best_model.pth preparing for further evaluation.

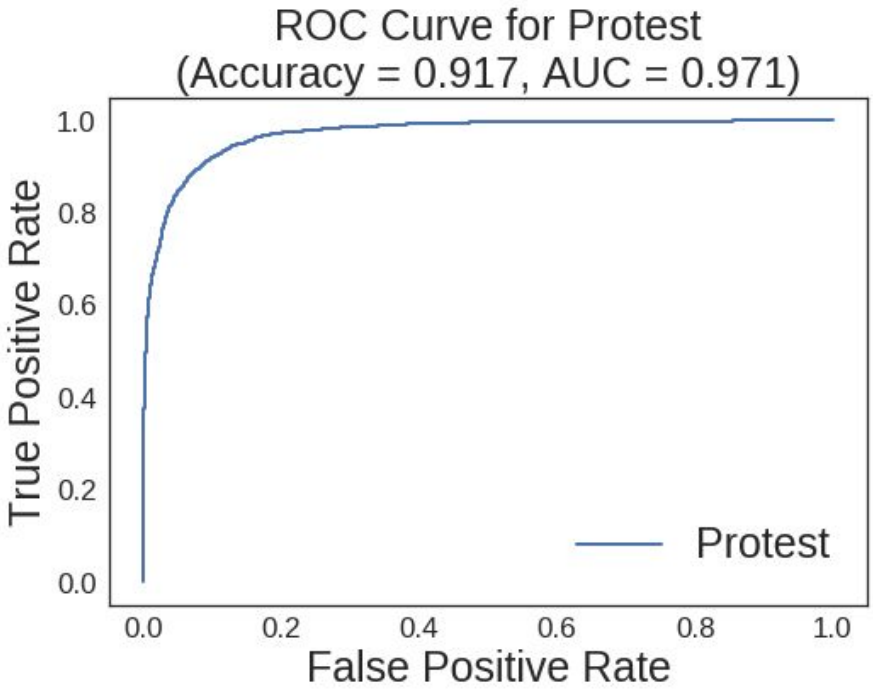
    As training epoch is increasing, train loss and val loss are both decreasing with lower speed as below.

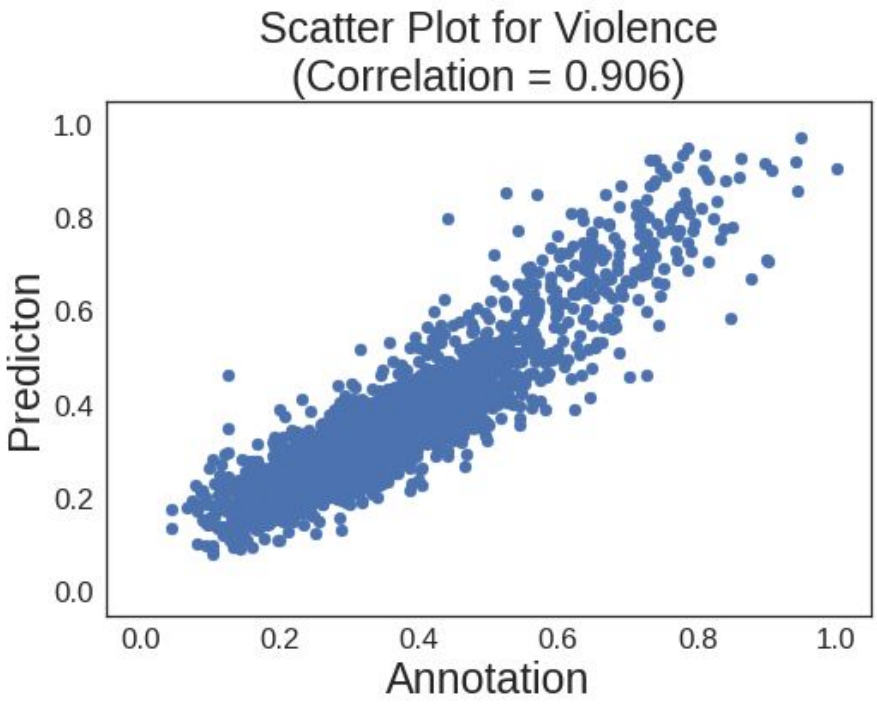


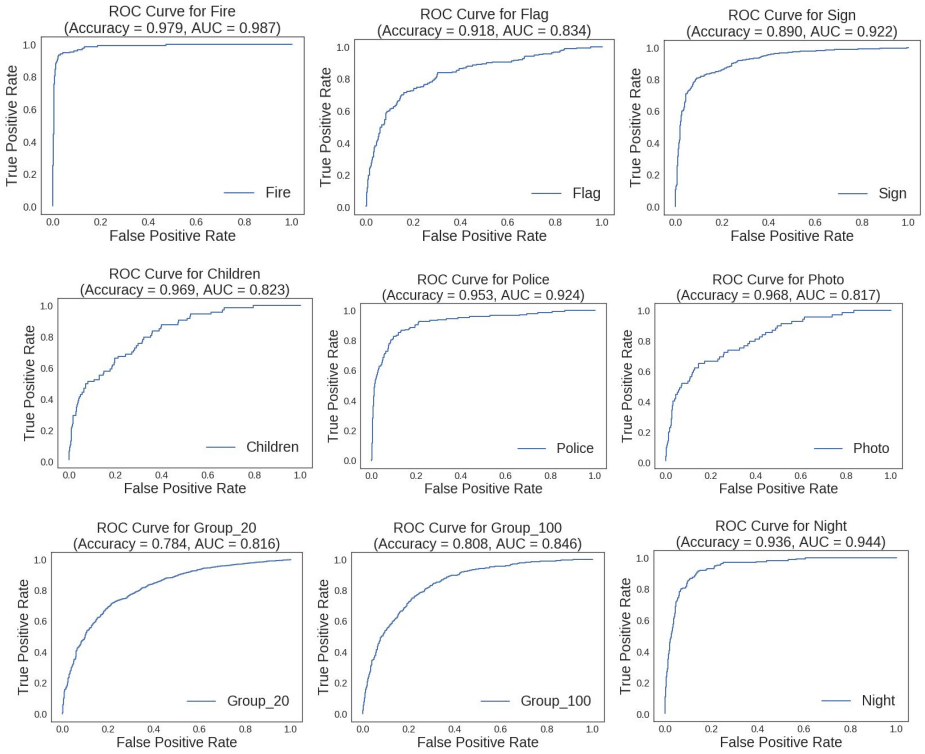
It can be concluded that, once epoch is larger than 50, the loss trend will fluctuate by the baseline at about 0.54 and 0.49, respectively for val loss and train loss.

- **Step 2:**
  We first cut images into specific format in util.py, do transformation such as resizing, rotation and normalization. Then read new crops into pred.py, apply best_model.pth to the PIL images, and return prediction results into a csv file.

- **Step 3:**
  Finally we run model-performance.ipynb as our model evaluation. For average accuracy, when iteration = 100, we get protest at 0.92, violence at 0.91 and other visual attributes at round 0.8-0.9.

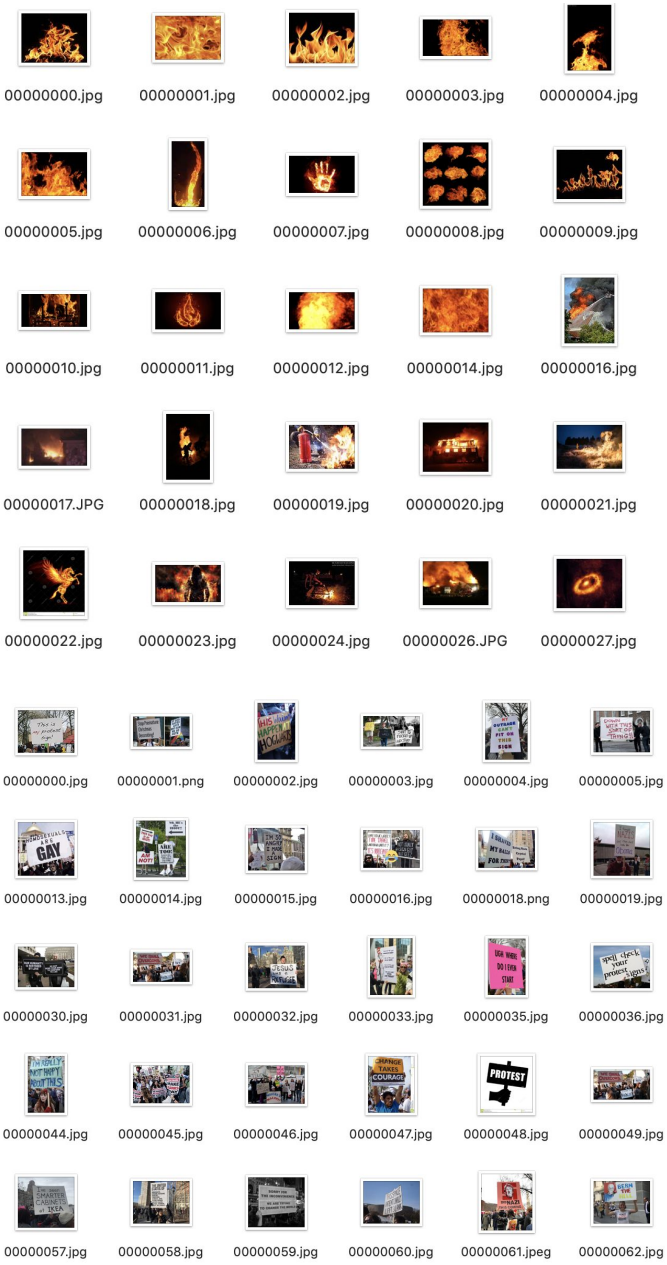

ROC Curve for Protest
(Accuracy = 0.917, AUC = 0.971)
True Positive Rate
False Positive Rate
Protest
0.0
0.2
0.4
0.6
0.8
1.0


Scatter Plot for Violence
(Correlation = 0.906)
Predicton
Annotation
0.0
0.2
0.4
0.6
0.8
1.0



III-2. Transfer Learning for Multi-labelled Classification

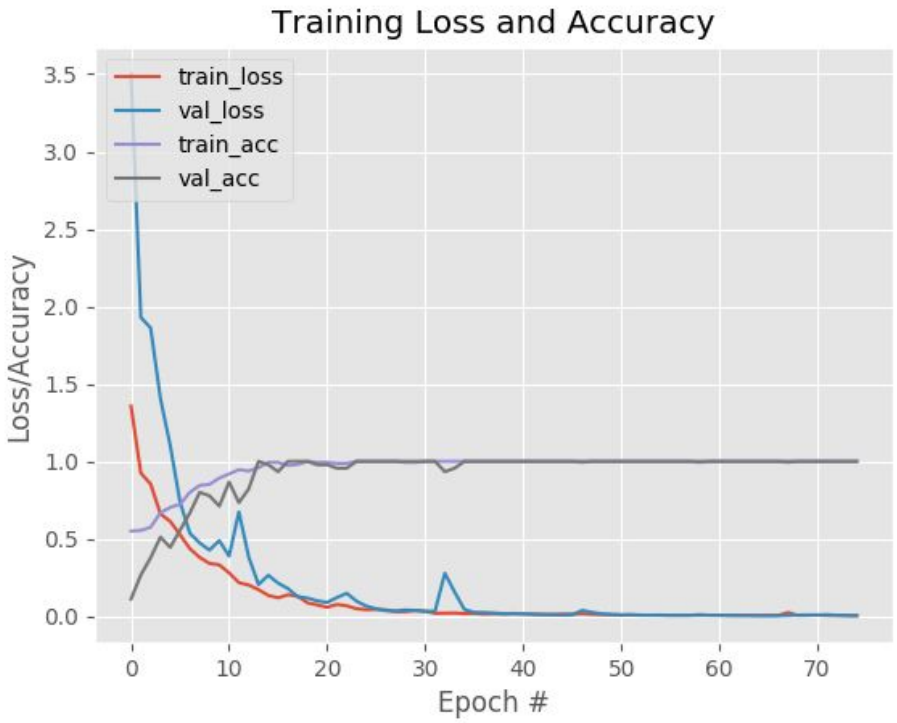
- **Goal:** Applying transfer learning technique to do multi-object detection without bounding boxes, using Keras library.
- **Input:** Pulled fire images using Bing Image API, a difficulty encountered is for fire there are lots of irrelevant images, ‘a gun fired’ is also considered as fire.
- **Steps:**
    - **Step1:** Train a single object classifier, such as fire/sign recognition. Examples from our training set (fire, protest sign) are listed below:



In step 1, our model is capable of recognizing a single object in the image with good true positive rate, but the false-negative rate is also high. It is likely that our training is not very comprehensive due to the fact that fire can differs in color, shape, with or without smoke, size. Sometimes an object with a similar color can also be identified as fire, so it is not easy to accurately classify if there is fire presented in the picture with a small size of raining set: hundreds of images.

- **Step 2:** Train a multi-object classifier. We used Bing Image API again to pull images. To simplify the procedure, we trained a classifier with fire and protest signs. The following images demonstrate the training process and the training plot.

```
Terminal
11/11 [==============================] - 8s 730ms/step - loss: 0.5740 - acc: 0.7748 - val_loss: 0.6711 - va
l_acc: 0.7944
Epoch 4/75
11/11 [==============================] - 8s 737ms/step - loss: 0.4279 - acc: 0.8313 - val_loss: 1.2104 - va
l_acc: 0.7667
Epoch 5/75
11/11 [==============================] - 8s 749ms/step - loss: 0.3418 - acc: 0.8649 - val_loss: 0.8003 - va
l_acc: 0.8111
Epoch 6/75
11/11 [==============================] - 8s 758ms/step - loss: 0.3358 - acc: 0.8710 - val_loss: 0.6738 - va
l_acc: 0.8000
Epoch 7/75
11/11 [==============================] - 8s 747ms/step - loss: 0.3588 - acc: 0.8520 - val_loss: 0.3809 - va
l_acc: 0.8389
Epoch 8/75
11/11 [==============================] - 8s 767ms/step - loss: 0.2147 - acc: 0.9117 - val_loss: 0.4102 - va
l_acc: 0.8611
Epoch 9/75
 3/11 [=======>......................] - ETA: 6s - loss: 0.4147 - acc: 0.8698
```



- The result generated did not meet our intended purpose when both fire and sign were both presented in an image. A possible solution is to modify our code to make sure that fire and sign can both be detected individually. Based on the output, it is likely that our classifier is only able to detect the dominating object, instead of both of them.

- **Conclusion and Analysis:**
  We found that current network is not able to perform multi-object detection very well. It can be used to analyze simpler pictures, for example, a picture of fire or a picture of signs. When these two elements are both presented, it can only classify either fire or sign, without identifying both of them.

  The reason we chose Keras was to avoid collecting dataset with bounding box. We also did some research on Yolo3, and tried to search images of fire coupled with bounding box data via Google Open Image Dataset v4. We still encountered the problem related to the dataset itself, since there didn't exist a dataset related to fire or protest signs. The available options such as fireplace, fire hydrant and traffic signs didn't fit our goal.

III-3. Transfer Learning for Single Class

For code and detailed report please see Jupyter Notebook file
transfer_learning_on_protest.ipynb

- **Goal:** Single label Transfer Learning using PyTorch, centering on 'Protest' without diving into details of image elements like the approach used in multi-label detection

- **Input:** Images from social media
- **Output:** ‘protest’ = 1 or 0 for input image
- **Tools used:** PyTorch, ResNet-18
- **Method:**
    - **Step 1: Pre-processing**
      To retrieve training and validation set from images in the dataset provided for this project. For code on this step, please see Jupyter Notebook file *pre-processing.ipynb*

      In pre-processing step, the following steps are performed:
        - Overview of training set images and corresponding labels
        - Filtering out null values
        - Locate columns of interest (protest), separate examples into positive and negatives
        - Locate images in folder news_imgs/train, partition and save into folders that contain positive and negative examples for training and validation steps

      Below shows a sample of augmented data used for training:

- **Step 2: Adapting Fine-Tuning** technique to train the convolutional neural network by replacing random initialization with a pretrained network, then enter training step using methods such as backpropagation, cost function minimization with Stochastic Gradient Descent.

  Brief discussion of result: in total 25 epochs were run, training loss had some overall fluctuations, but reaching last epoch, the training loss is at its minimum, and accuracy is at its highest. Best Validation Accuracy achieved with this model is 0.868750. Training completed in 42 minutes on a CPU.

  Sample Result:

predicted: no_protest

predicted: no_protest

predicted: protest

predicted: protest

predicted: no_protest

predicted: protest

- **Step 3: Using CNN as a fixed feature extractor** by freezing the weights for all the network except ones of the final fully connected layer. The weights in the last layer is replaced with new random weights and only this layer is trained.

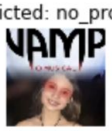
predicted: protest

predicted: protest

predicted: no_protest

predicted: no_protest

predicted: no_protest



predicted: protest

- **Comparison of the two Transfer Learning approaches:** the Fine-Tuning approach has overall lower training loss and reached slightly better accuracy than the approach using CNN as a fixed feature extractor. Thus for the improvement section, improvement on performance will be focused on the fine tuning approach. Please see details in the improvement section.

  Below examples of performance for the last 4 epochs from the two approaches are shared:

  - Fine Tuning: overall best validation accuracy = 0.8688

```
Epoch 21/24
----------
train Loss: 0.3957 Acc: 0.8325
val Loss: 0.3727 Acc: 0.8375

Epoch 22/24
----------
train Loss: 0.3578 Acc: 0.8525
val Loss: 0.3750 Acc: 0.8313

Epoch 23/24
----------
train Loss: 0.4369 Acc: 0.7950
val Loss: 0.3850 Acc: 0.8187

Epoch 24/24
----------
train Loss: 0.3347 Acc: 0.8625
val Loss: 0.3864 Acc: 0.8250
```

- Fixed Feature Extractor: overall best validation accuracy = 0.8375

```
Epoch 21/24
----------
train Loss: 0.4523 Acc: 0.7775
val Loss: 0.3925 Acc: 0.8063

Epoch 22/24
----------
train Loss: 0.5156 Acc: 0.7450
val Loss: 0.3884 Acc: 0.8250

Epoch 23/24
----------
train Loss: 0.5146 Acc: 0.7650
val Loss: 0.3834 Acc: 0.8125

Epoch 24/24
----------
train Loss: 0.5407 Acc: 0.7250
val Loss: 0.4078 Acc: 0.8250
```

IV. Improvement

We approached improvement from 3 perspectives: **before training, during training and after training**. Also, our approach of trying to solve a problem through different angles is also part of our improvement, we found this trail and learning process very rewarding.

For our approach on reproducing the method described by the paper, we mainly targeted improvement before training, focusing on **dataset quality**. We emailed paper author requesting

the original dataset (UCLA_PROTEST_IMAGE_DATASET) used by their paper, and used it as our input. It's an improvement because the dataset contains more annotated categories, particularly the 'Group 20' and 'Group 100', it's a useful perspective in reaching our goal on determining the emergency level given a protesting event, assuming the larger group of people the more effort required to bring situation back to normal.

During training, we tried for **parameter tuning**, such as tuning on learning rate, step size, momentum to better fit our input data; however, since the models are built upon pre-trained CNN, and the most optimum weights are carried over, we assume that performance are at an optimum level to begin with, so if there're improvement, it's likely not going to be significant unless we dive deep into the model and tune specific for our task, but it's beyond the scope here. This approach is tried on the Transfer Learning for Single Class - Fine Tuning model. Result of the parameter tuning is as expected, the original setting (lr=0.001, momentum=0.9, step_size=7, gamma=0.1) turns out to be the most optimal and carried out with PyTorch optim.sgd class. The only improvement is speed achieved from configuring the model to be deployed with Cuda, originally it took 42m11s to complete 25 epoches, now it takes only 56s.

Our second approach on improvement is **building extra functionalities**, and if time's given, integrate them on to one of the baseline approaches. We're hoping this part would count towards extra credit for creativity. For this one, we tried two methods, one is the **Expert System** mentioned in Problem Definition section, which can be added to the paper reproduction result, as this one generates a violence ratio that can be further analyzed with the Expert System. Please find details below.

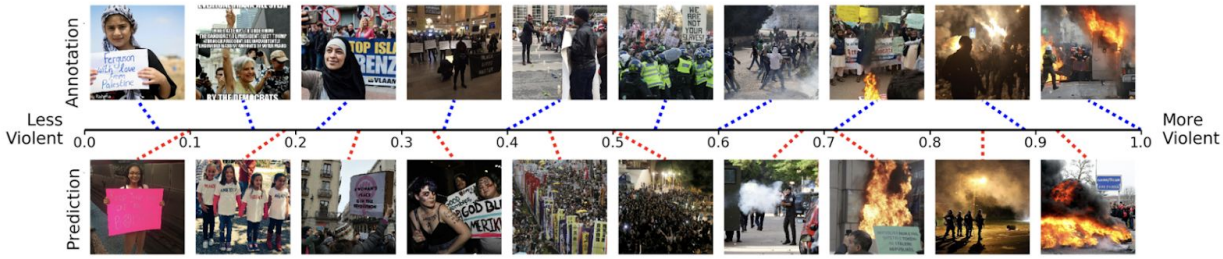
Second one is **Crowd Detection**, following same logic as improvement before training approach *(at time of writing and deciding on improvement, we do not know if the paper reproduction is going to succeed, but we want to get an estimate on number of people participating in a protest event as it helps in determining emergency level, so we decide to add this extra functionality)*. For example, a detection of number of people in a crowd could be added onto the transfer learning for protest detection model, so the end result would be 'an image shows a protest is taking place, and the size of the protest is small/medium/large'.

IV.1- Building Extra Functionalities - Expert System

A simple expert system model to illustrate our idea.

- Code can be found in file simple_expert_sys.py
- **Input:** output from deep neural network, a .csv file containing predicted labels for all categories (ex: protest, violence, fire etc.)
- **Output:** A emergency level from Low, Medium, High
- **Method:** Here, the emergency level is solely depended on violence score (a column in .csv file)

Based on this chart we found from the original paper[9] for violence ratio:



Emergency Level's categorized as follow, intervals are lower and upper bounds for violence ratio:

- Low: [0.0, 0.3]
- Medium: (0.3, 0.7)
- High: [0.7, 1.0]

- **Possible Improvement given more time:**

1. Involving more input measure for generating emergency level, such as presence of fire, number of people involved etc.
2. Have more detailed categorization for emergency level

IV.2-Building Extra Functionalities - Crowd Detection

For a protesting event, it's helpful to know the number of participants, because it determines the scale of protest and methods for bringing the situation back to normal hugely varies because of it; knowing the crowd count will help our system to generate better estimation of emergency level; thus we turned our attention to extract this information from a given image. We tried two different possible implementations.

First one is called CSRNet-keras, it's implemented in keras-tensorflow. It has advantage in deployability, it was used on the ShanghaiTech dataset. This dataset consists of images with both high density of crowd and low density of crowd. Through data preprocessing, the images can be converted to density maps. After performing head counting, this model can calculate number of people within the images.

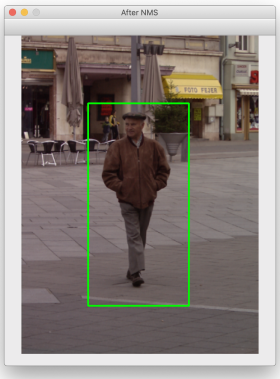
This method was quite promising at first glance, but we quickly found that it required us to manually create files containing ground truth data. Which would took more time than we can allocate, so we switched to another approach.

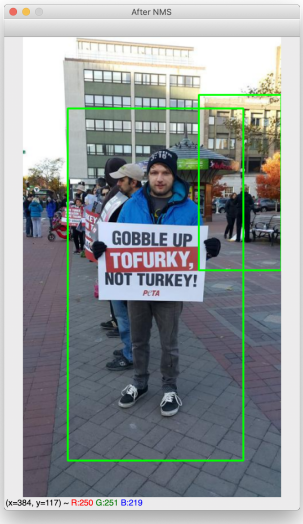
---

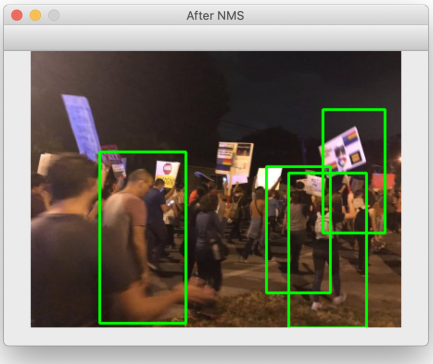
[9] Donghyeon Won, Zachary C. Steinert-Threlkeld, Jungseock Joo. 2017. Protest Activity Detection and Perceived Violence Estimation from Social Media Images. In Proceedings of the 25th ACM International Conference on Multimedia 2017.

For the second approach, We found that OpenCV has its own built-in methods to use pre-trained model to perform pedestrian detection. We followed online resources and built the detection model. The result was good when an image captures the entire frame of a person, but its performance degrades when people frame(s) were not completely captured. Below are some examples generated by our code, showing bounding boxes around detected figures.

We can see that this approach is ideal for pedestrian detection, while less ideal for protest scenarios since it cannot detect overlaid figures. The first two pictures have accurate bounding boxes, but the model performance worsens on the other two samples containing more overlaid figures.







V. Conclusion and File Structure

Above are the methods we tried and their results.

All code and files can be found on GitHub with link:
https://github.com/JerryWu96/AI_CS640_BU/tree/master/News%20Image%20Classification

File Structure in our repository is as follow:

- Baseline
    - Transfer Learning on Multi Object Detection
    - Transfer Learning on Single Class
    - Paper Reproduction
- Files
    - Brainstorming Notes
    - Report
    - Presentation PPT
- Improvement
    - Crowd Detection
    - Expert System

Thank you for your time.