Finding Protests in Social Media Data using CNNs and Transfer Learning

Benjamin Zhou
Stanford University
Department of Computer Science
bzhou2@stanford.edu

Gaspar Garcia Jr.
Stanford University
Department of Computer Science
gaspar09@stanford.edu

Dylan Moore
Stanford University
Department of Computer Science
dmoore2@stanford.edu

Abstract

In this paper, we build various classifiers and compare their performance in discriminating social protests from non-protests in China, a problem with few previous results. We use a labeled dataset of 500,000 protest and non-protest images taken from the Chinese social media site Weibo. Our work includes a baseline Support Vector Machine classifier, 3-layer and 5-layer convolutional neural networks, and transfer learning models based on SqueezeNet and VGGNet. On the test set of 4400 random images, our best model achieves 67% accuracy.

1. Introduction

Determining the stability of authoritarian regimes, among them China, Russia, Iran, and Saudi Arabia, is of central importance to policy makers, businesses, societal actors, and scholars around the world. For many authoritarian regimes, protests and visible civilian unrest pose a serious threat to the regime's stability.

Independent measures of protest activity in such countries would be valuable for scientific and public policy purposes. However, because repressive regimes actively remove evidence of protests, there are very few publicly available metrics for this issue.

The Chinese government will suppress social protest using force if it becomes too large. Weibo is a social media platform in China, where users can tweet or post photos about their life and surroundings. In an event of social protest, there may be various photos or statuses posted from user accounts. If the photos do not become too trending, the government will not remove them. An account called wickedonnaa, which were a husband and wife documenting social protest in China before being censored, gathered a dataset of $\sim$ 500,000 social media posts. Professor Jennifer Pan of Stanford's Department of Sociology and Han Zhang, a Sociology PhD student at Princeton, provided us with their dataset. Given the inavailability of social protest statistics in China, the goal of our project is to build classifiers that can discriminate between protest and non-protest images well.

2. Related Work

There was not much literature on our particular problem. Previous work on this classification problem was by Han Zhang, a PhD student at Princeton. Han's work tried a simple two layer CNN classifier with a batch of 20,000 images. Han's CNN was evaluated on the test dataset described below in section 3 and had precision of approximately 40% for protest samples.

Previous results in crowd counting provided us some insight on how to classify protests. The research of Lawrence, Steve, et al., Wang, Chuan, et al., and Yingying Zhang, et al. suggested that CNNs might ultimately be the most effective classifiers for social protest. Their results concluded that traditional hand-crafted methods such as SIFT and HOG might be ineffective in classifying social protest, since social protest images can be characterized by images with very dense crowds of people. As the density of crowds grows, SIFT features would be less ineffective, as only a relatively few amount of pixels would be dedicated to each person. That would make it harder to perform classification with SIFT features. Inspired by Xia, et al., we implemented an SVM classifier as a baseline result, in order to compare linear classifiers with CNNs.

Figure 1. Above are 16 sample images from our dataset. The left eight images are non-protest, and the right eight images are protest.

3. Dataset and Features

The dataset from Weibo consists of 231618 protest images and 261516 non-protest images. Protest images are labeled as 1, and non-protest images are labeled as 0. The size of the training data is about 10 GB, and images had width and height rescaled to 100x100x3, where the last dimension corresponds to the RGB depth. The test data set has 4400 images, which were randomly selected from a target dataset of 4 million images from Weibo. In the test set, about 10% of the samples are protest related images. The dataset was uploaded onto Dropbox.

From the above images, we see that the protest images vary in the number of people they contain. The first six protest images do not contain a high density of people, in contrast to the last two images. Protest images also likely contain people holding signs, police, or violence. Non-protest images can contain essentially anything.

4. Approach

The Chinese government defines social protest as any form of collective action that contains at least three people. If our learning was not supervised, we could try to approach this problem by counting the number of people in an image to classify it as protest or non-protest. However, just relying on people count as an indicator for protest can lead to a high number of false positives, since that indicator would be too loose. If the images also contain police or violence, then it is more likely that the images are indeed protest.

However, since the dataset came labeled, we can do supervised learning. The following subsections describe various classifiers that we experimented with. We first implement an SVM as a baseline, go on to implement CNNs, and finally transfer learning with VGGNet and SqueezeNet.

One thing to note is that in our classification problem, the test data is likely generated from a different distribution than the training data. The test data contain random images taken from 4 million images off Weibo. The majority $\sim$90% of them are non-protest, which can involve many different things, and makes the test dataset imbalanced. This inspired us to use transfer learning, i.e. borrowing existing neural networks trained on ImageNet which are able to capture many different things.

4.1. Training

All our work is done on a Google cloud instance using Jupyter notebooks. We use Scipy for our shallow models and our deep models are all written in TensorFlow.

For both models described below, we experiment with a mini-batch of 20,000 training images from our dataset. These samples were randomly chosen, where half of them are protest images and the other half are non-protest images. Our validation set was a set of 2,000 different images with the same 50/50 distribution of labels. We used the cross entropy loss function for classification in the CNNs as shown below. The cross entropy loss for a particular training example i is,

$$L_i = -\log(\frac{\exp^{f_{y_i}}}{\sum_j \exp^{f_{y_j}}})$$

where f_{y_i} refers to the output score for the true label of training example i. The entire cross entropy loss is then defined as,

$$L = \sum_i L_i$$

4.2. SVM baseline

We first fit an SVM over flattened images with a linear kernel. The SVM perfectly fitted the 20,000 training images and achieved 75% accuracy over the 2,000 sample validation set.

4.3. 3-Layer CNN

Our first CNN model uses 3 convolutional layers followed by two affine layers. The first conv layer uses 32 4x4 filters with stride 2. The second conv layer uses 64 4x4 filters with stride 2. The third conv layer uses 64 4x4 filters with stride 1. Each convolutional layer uses relu activation and a batch normalization layer. The output from the last convolutional layer flows into a 2x2 max pool with stride 2. The max pool output is fed to an affine layer with 1048 outputs. This layer is followed by a dropout layer with dropout

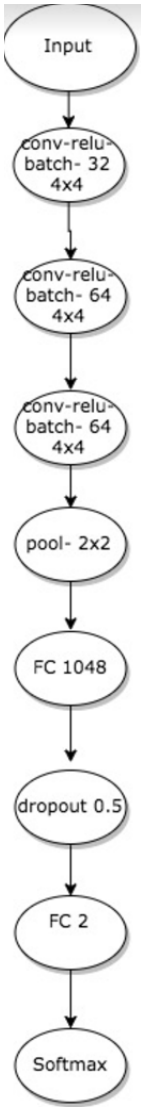

Figure 2. Architecture of 3-layer CNN.

probability 0.5. The final layer is an affine layer with 2 outputs, one for each protest and non-protest class.

We used an Adam optimizer and L2 regularization over the affine weights. The regularization rate is 5e-2 and the learning rate is 5e-4.

4.4. 5-layer CNN

For our 5-layer CNN, the first conv layer uses 32 3x3 filters with stride 1. The second conv layer uses 64 4x4 filters with stride 2. The third conv layer uses 64 4x4 filters with stride 2. The fourth conv layer uses 128 3x3 filters with stride 1. The fifth conv layer uses 128 3x3 filters with stride 1. Each conv layer uses relu activation and a batch normalization. The output of the last convolutional layer goes into a max pool layer, which uses filters of size 2x2 with stride 2. This layer is followed by a fully connected layer with 2048 output units, then a dropout layer with dropout probability 0.5, and finally a fully connected layer with 2 output units.

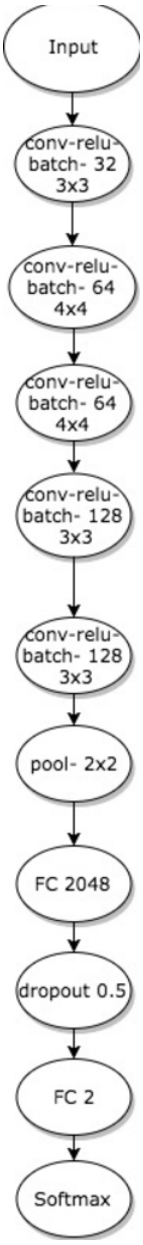

Figure 3. Architecture of 5-layer CNN.

We used an Adam optimizer and L2 regularation over the affine weights. The regularization rate was 5e-2, and the learning rate was 5e-4.

4.5. Transfer Learning with SqueezeNet

We implemented transfer learning by using the existing neural network SqueezeNet, which was trained on ImageNet. We added our 5 layer neural network to the top of the 3rd layer of SqueezeNet, i.e. we fed our protest training data into SqueezeNet, and used the extracted features from

the 3rd layer to then feed into our 5-layer neural network. We chose the 3rd layer because it achieves the highest accuracy, and also it represents high level features of the protest images.

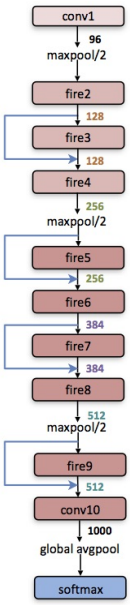

Figure 4. Architecture of neural network SqueezeNet used for transfer learning. We use the extracted features from the third layer fire3 to train our 5-layer CNN described earlier.

We used an Adam optimizer and L2 regularization over the affine weights. The regularization rate was 5e-2, and the learning rate was 5e-4.

4.6. Transfer Learning with VGG

We implemented transfer learning using VGGNet as shown in Figure 5, which is trained on ImageNet. We extracted the features from the second to last fully connected layer of VGGNet, which contained 4096 output units, and fed them into our own fully connected layer of two output units.

We took these extracted features and fed them into our own fully connected layer of 2 output units.

5. Experiment Discussion and Results

In this section, we describe our results and the motivation behind our experiments.

Figure 6 shows the training accuracies, validation accuracies, target or test accuracy, precision, recall, and number of epochs trained statistics of the various classifiers.

The baseline SVM classifier achieves 75% validation accuracy and 53% target accuracy. The 3-layer CNN and a deeper 5-layer CNN both achieve higher validation accuracies at 85% and higher target accuracies at 62%. Our

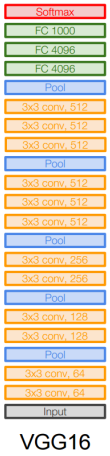

Figure 5. Architecture of neural network VGG16 used for transfer learning. We take away the last layer, and replace it with our own fully connected layer of 2 units.

transfer learning models seemed to be the most promising. We trained them for less epochs than regular CNNs, but achieved similar if not higher accuracies. The transfer model using VGGNet achieved our highest test accuracy at 67.6%. In all models, the precision was relatively low, compared to Han's previous work of 40% precision. Ultimately, our goal is to build a convolutional neural network that can achieve $> 90\%$ accuracy on our test data with high precision as well.

	Train Acc.	Val Acc.	Target Acc.	Recall*	Precision*	Epochs
SVM Baseline	1	0.75	0.53	0.1	0.19	-
3-Layer CNN	0.91	0.85		0.22	0.23	15
5-Layer Deeper Neural Network	0.88	0.85	0.65	0.55	0.22	15
Transfer Learning SqueezeNet	0.89	0.86	0.62	0.72	0.24	5
Transfer Learning VGG16	0.917	0.882	0.676	0.785	0.269	5

Figure 6. Statistics for the various classifiers.

For our 3-layer CNN, we initially had one affine layer with 2 outputs. The convolutional layers used 7x7 filters, 4x4 filters and 3x3 filters all with stride 1. Every conv layer used relu activation and batch normalization. However, this yielded a low training accuracy with a very noisy, yet flat lost curve, indicating that the network was not learning, shown in Figure 7.

In response, we experimented with the architecture by adding another Fully connected layer before the final affine output. This new affine layer had 128 outputs. The layer increased the training accuracy dramatically and with tuned learning rates, we easily overfitted the train data.

This network had validation accuracy at $<79\%$ and train accuracy up to 95%. In order to counter the overfitting model, we added L2 regularization over the weights on the

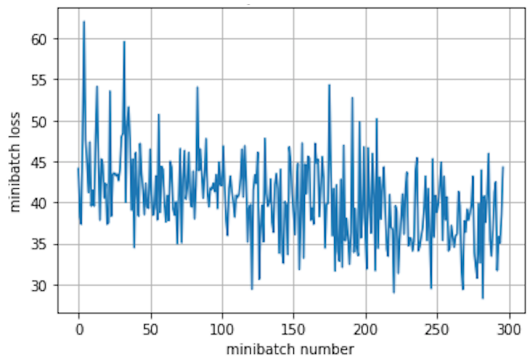

Figure 7. Initial noisy loss curve in the 3-CNN training.

affine layers and added a dropout layer after the second to last affine layer.

At this point the hidden layer had 128 outputs and wasnt learning so well on training data. We took inspiration from the architecture by AlexNet and VGG16 which use hidden affine layers with 4096 outputs preceded by 2x2 max pool. We added a 2x2 max pool after the final conv layer and increased the outputs for our affine layer to 1048. At this point we had an overfit issue again. We tuned hyperparameters over 3 epochs. We performed a random grid search to tune the learning rate and regularization rate between 1e-2 and 1e-7. Dropout probability was tuned linearly between 0.1 and 0.5. The loss curve for the tuned model is shown below.

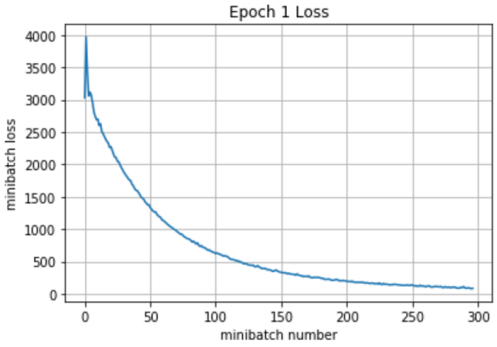

Figure 8. Final loss curve for the 3-CNN.

The loss curve suggests that the network learned well. Our architecture achieved 81% train accuracy and 81% validation accuracy with 3 epochs. We increased the number of epochs to 15 to get a final result of 92% train accuracy and 85% validation accuracy.

In addition, we wanted to experiment with a deeper network. We used 5 layers and tuned them in a similar way as in the 3-layer CNN. But the resulting accuracies were similar, and hence we decided to turn away from deepening the network.

We also visualized the learned weights after the 1st convolution layer of the 3-CNN as shown in Figure 9. The weights are quite coarse since they are viewed in 3x3, the dimension of the filters.

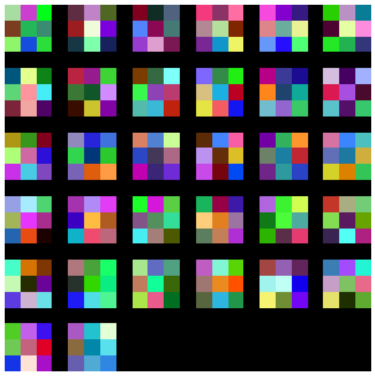

Figure 9. Visualization of the first convolutional layer weights of the 3CNN (3x3)

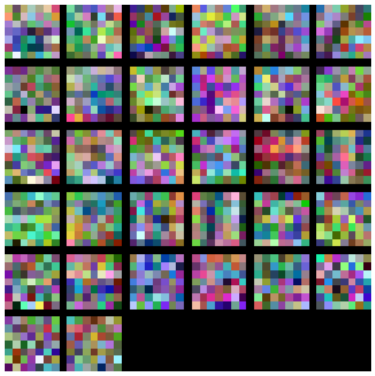

Figure 10. Visualization of the first convolutional layer weights of the 5CNN (7x7)

Figure 10 shows the learned weights after the 1st convolution layer of the 5CNN, but supposing the filters had dimension 7x7. We ran this just to see how well our network was learning from the data itself, and to make the visualization of the weights less coarse. In some of the weights such as the top right, the network seems to be detecting information such as edges.

Finally, we turned to transfer learning models. Our main reasoning for using transfer learning was that we thought existing neural networks trained on ImageNet would be able to recognize non-protest images better, which consisted of 90% of the test data. It turned out that the transfer models did the best, with the VGG16 model achieving the highest accuracy of 67% on the test set. Also, our transfer learning models achieved a much higher recall than the previous models. This suggests that the transfer learning models do a much better job at classifying protest images as protest.

6. Challenges and Future Work

The experiments described in this milestone use a subsample of 20,000 from 500,000 images in the whole dataset. We tried this number of training samples initially as Han's previous work suggested. But this is a small fraction of the data. Moving forward, it would be beneficial to use more images for both training and validation. We haven't yet reached our target accuracy of $> 90\%$, but training on more non-protest images would make our network more robust.

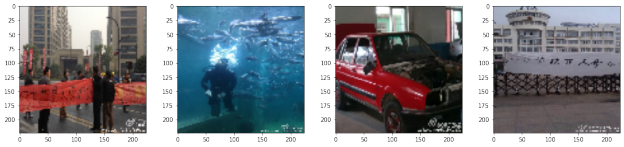

Figure 11. Images classified by the VGG16 transfer learning model as TP, TN, FP, FN, respectively.

The precision of our models is also low at $< 30\%$. This is worse than Han's previous precision of 40%, but there is much room for improvement. The figure above shows examples of images classified by the VGG16 transfer learning model as TP, TN, FP, and FN. The FP image is interesting; the classifier classified a car as protest, which suggests the model could be not robust. The FN image is understandably difficult to classify as protest, since there are no actual people protesting in the image, but just their protest sign.

Future work could include incorporating other metrics to measure size or intensity of protest. It is valuable if we can also determine whether images of protests are violent or non-violent.

7. Conclusion

We want to thank Professor Jennifer Pan of the Department of Sociology, and Han Zhang for advising us on this project. Furthermore, we want to thank the staff of CS231A for leading an interesting and informative class.

Below is a link to the github of our code.

https://github.com/gaspar09/EyesOnProtests

8. References

1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026-1034

2. Wang, Chuan, et al. "Deep people counting in extremely dense crowds." Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015.

3. Lawrence, Steve, et al. "Face recognition: A convolutional neural-network approach." IEEE transactions on neural networks 8.1 (1997): 98-113.

4. Xia, Jiantao, and Mingyi He. "High Dimensional Multi-spectral Image Classification by SVM and Its Characteristic Analysis [J]." Computer Engineering 13 (2003): 009.

5. Levis, Joel, et al. "Joint Deep Exploitation of Semantic Keywords and Visual Features for Malicious Crowd Image Classification." arXiv preprint arXiv:1610.06903 (2016).

6. Yingying Zhang, et al. "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network"