Detecting the Use of Hand Hygiene Dispensers: Research Sample

Edward Chou

1. Overview

For this research sample, I decided to tackle the classification task from multiple angles to explore and compare the approaches that can be used for this dataset. I attempt to answer two questions: Which modeling approach is best suited for the classification task of whether a hand sanitizer was being used or not, and which is better between training one model for all the camera data versus training an individual model for each camera. For training, I altered between using MacBook Pro 2.8 GHz quad core for the SVM training, and standard FloydHub hardware (Tesla K80 GPU, Intel Xeon Dual Cores) for CNN training.

At the conclusion of my experimentation, I found that the answer to the first question was transfer CNN with augmented data, which produces an average of 0.9555 validation accuracy for each camera model. I also found that training one weighted transfer CNN model on all the data performs better than training a weighted transfer CNN model for each camera, although this result could differ with augmented data.

2. Data

Because I decided to use VGG16 models (image classification networks) for transfer learning and generating CNN features along with HOGs for SVM training, I decided that representing the data as 'ocean' images was the most appropriate. For all model training and testing, the data is loaded and converted into an image as specified in the proposal under the Visualizations section, and resized to (224 x 224) as the preferred VGG16 format.

I subsampled 2640 images for each camera model, using a 66:33 split for training and validation sets. I excluded data from camera numbers 10, 15, 21, 22, and 24, because they had no images labeled positive for hand-sanitation use. The remaining camera models varied greatly in negative to positive label ratios, ranging from over 200:1 negative to positive imbalance to a near 1:1 split between the labels. To test transfer CNN with data augmentation, I randomly apply several transformation techniques altering the rotation, translation, shearing, and brightness. These images, along with the originals, are saved to a separate dataset I use specifically for training the transfer CNNs on augmented images.

Figure 1: Transformed Images

3. Methods

3.1. Different Binary Classification Models

3.1.1 SVM with HOG

HOGs, or histogram of oriented gradients, is a edge detector that captures information including the contour and silhoutte of the image. After performing HOGs on every image, I reduce the dimensionality of the features to 200 using Principle Components Analysis to reduce variance and overfitting. I set the weights of the class of the SVM by calculating the ratio of negative to positive labels and setting the weight of the positive label to be the ratio.

3.1.2 SVM with CNN Features

Next, I used the weights of the hidden layer after running each image through a VGG16 as features, as outlined here. CITE. The output layer is 512 7x7 feature maps, which I reshape and compress to a dimensionality of 200 using PCA to reduce variance and overfitting.

3.1.3 Weighted Transfer CNN

I try a transfer learning technique, using the VGG16 network, removing the softmax and dense layers, adding my own fully connected layer, and setting the rest of the layers to untrainable. I set the weight of the positive label to be ratio of negative to positive labels. The learning rate is set to 0.0001, the minibatch size to 64, and the epochs to 30.

3.1.4 Transfer CNN with Augmented Data

Finally, I use the same transfer learning network, and train on the transformed image dataset. I do not set the weight this time, as the ratio of positive and negative labels should be close to 1:1 after the data augmentation. The learning rate and minibatch size are set the same, and I reduce the epochs to 10, due to the fast training speed.

3.2. Training One Model vs. Multiple Models

3.2.1 One Weighted Transfer CNN

I use the same VGG16 transfer learning network as before with learning rate is set to 0.0001, the minibatch size to 64, and the epochs to 30, and train on data from all the cameras with the sanitation vs non-sanitation labels, using the labels ratio as the class weight.

3.2.2 Transfer CNN models trained for each camera

I create a VGG16 transfer learning network for each camera, and train on data for that camera specifically, using the same parameters as before and the labels ratio as the class weight. I also create V66 transfer learning networks for each camera trained on the augmented images without class weights. Finally, I also train a camera classification model using the VGG16 transfer learning network, which takes in an image and attempts to classify which camera its from.

4. Results and Discussions

4.1. Comparing Binary Classification Models

I pick camera 08 to test the four models. The relevant tables are Tables 1, 2, 3, and 4. Overall, the best performance comes from the transfer CNN on the augmented data. SVM on CNN features performs better than SVM on HOG features and surprisingly weighted transfer CNN. Both Tables 1 and 3 show that precision and recall for class 1 is zero, showing that the HOG SVM and the weighted transfer model aren't well suited for training this dataset.

4.2. Training One Model vs. Multiple Models

For the single model trained on all data, we get train accuracy: 0.7342 and test accuracy: 0.7334. The accuracies for each camera is recorded in Table 5, where WTrA is Weighted training Accuracy, WTeA is weighted testing accuracy, AugTrA is augmented training accuracy, and AugTeA is augmented testing accuracy.

Overall, we can see that the average accuracy of weighted transfer learning is lower than that of the one model, which is surprising. It is possible that the larger amount of data helps improve the performance of the single model. The augmented models outperform the one model, but I didn't have time to train a single model on all the augmented data to compare.

Finally, I found that the camera classification model achieves training accuracy of 0.8859 and a testing accuracy of 0.9320. Therefore, I conclude that training individual camera networks will work reasonably well even without camera labels.

Class	Precision	Recall	F1-Score	Examples
0	0.90	1.00	0.95	594
1	0.00	0.00	0.00	66
Avg/Total	**0.81**	**0.90**	**0.85**	**660**

Table 1: Classification report for HOG SVM

Class	Precision	Recall	F1-Score	Examples
0	0.99	0.95	0.97	594
1	0.70	0.94	0.80	66
Avg/Total	**0.96**	**0.95**	**0.96**	**660**

Table 2: Classification report for CNN features SVM

Class	Precision	Recall	F1-Score	Examples
0	0.89	1.00	0.94	1766
1	0.00	0.00	0.00	214
Avg/Total	**0.80**	**0.89**	**0.84**	**1980**

Table 3: Classification report for weighted transfer CNN

Class	Precision	Recall	F1-Score	Examples
0	1.00	0.98	0.99	1766
1	0.89	0.98	0.98	214
Avg/Total	**0.99**	**0.98**	**0.98**	**1980**

Table 4: Classification report for transfer CNN on augmented data

Camera	WTrA	WTeA	AugTrA	AugTeA
02	0.7455	0.7710	0.9948	0.9601
04	0.7196	0.7678	0.9941	0.9844
06	0.6931	0.6812	0.9834	0.9414
08	0.8229	0.8721	0.9922	0.9762
11	0.7587	0.7403	0.9955	0.9691
15	0.6674	0.6315	0.9992	0.9989
23	0.7732	0.7809	0.9825	0.9409
39	0.6425	0.7394	0.9975	0.9921
52	0.6977	0.6536	0.9850	0.9425
59	0.6793	0.7027	0.9982	0.9872
62	0.7461	0.7516	0.9877	0.9222
63	0.6594	0.6480	0.9753	0.9216
72	0.6253	0.6163	0.9770	0.8849
Average	**0.7101**	**0.7197**	**0.9894**	**0.9555**

Table 5: Regression results for the bikes dataset.

5. Future Work

If I had more time, I would finish some techniques I planned to implement, including training an entire model on all the augmented images and linking the camera classification model to each individual camera binary classification network.