Real Time Monitoring of CCTV Camera Images
Using Object Detectors and Scene Classification for Retail and Surveillance Applications

Anand Joshi
CS229-Machine Learning, Computer Science, Stanford University, CA
aaj1031@stanford.edu
Fall 2017

Abstract **- With the advent of object detection pipelines, we should be able to use their output to supervise more complex ML tasks, like recognizing specific scenes and performing complex image queries. Specifically, it would be interesting to see how we can use Deep Learning (DL) and especially Convolution Neural Networks (CNN) to build complex scene detection algorithms. Different CNN architectures and frameworks will be evaluated and customized in the course of the project and the best choice will be used for the purpose of application.**

1 Introduction

Current surveillance and control systems in retail and elsewhere, still require human supervision and intervention. This work will try to provide a detection system in videos on real time basis, appropriate for; surveillance and control, inventory tracking, theft deterrence, threat perception and detection etc. and apply Machine Learning/Deep Learning techniques for real world applications. This will try to automate many tasks, which can be error prone otherwise due to human errors and fatigue. This solution can potentially have capability to provide real time alerts, notification on smart phones/tablets and provide rich data for analytics purpose.

2 Relevant Previous Work

Convolutional Neural Network on Image Classification:
In object classification, CNNs have become extremely popular due to their high success rates in accurately recognizing objects. In pattern and image recognition applications, the best possible Correct Detection Rates have been achieved using CNNs.
In 2012, Krizhevsky and et al.[1] trained a deep convolutional neural network to classify images in LSVRC-2010 ImageNet into 1000 kinds of classes with much better precision than previous work, which marked the beginning of usage of deep learning in computer vision. AlexNet has 60 million parameters and 650,000 neurons, consists of five convolutional layers. Those layers are followed by max-pooling layers, and three globally-connected layers with a final 1000-way softmax layer. After that, there are several symbolic milestones in the history of CNN development, which are ZFNet by Zeiler and Fergus, VGGNet by Simonyan et al., GoogLeNet (Inception-v1) by Szegedy et al and ResNet by He et al. GoogLeNet or Inception V1 was the winner of ILSVRC 2014. It largely reduced the ImageNet top-5 error from 16.4% which obtained by AlexNet to 6.7% . The Inception deep convolutional architecture was introduced, with the advantages of less parameters (4M, compared to AlexNet with 60M) . Average Pooling instead of Fully Connected layers at the top of the ConvNet was applied to eliminate unnecessary parameters. Later, there are several more advanced versions to Inception V1. Batch normalization was introduced in Inception V2 by Loffe et al. Later the architecture was improved by additional factorization ideas in the third iteration which will be referred to as Inception V3. Inception V4 has a more uniform simplified architecture and more inception modules than Inception V3. Szegedy et al. designed Inception-ResNet to make full use of residual connections introduced by He et al. in and the latest revised version of the Inception architecture. Training with residual connections accelerates the training of Inception networks by utilizing additive merging of signals.

Deep Learning Frameworks:
In 2014, Jia and et al.[2] created a clean and modifiable deep learning framework: **Caffe**. Just a year ago Google Brain Team open sourced another deep learning framework called **TensorFlow**, for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. Another such framework worth mentioning is **Torch** developed by NYU and Facebook.

3 Modeling and Algorithm

CNN is made up of a series of convolutional and pooling layers, in which in the final layer is fully connected at the output neurons with a softmax layer to give the confidence of the prediction (Fig. 2.).

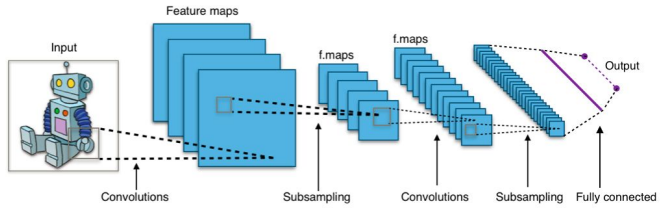


Fig. 1. General architecture of Convolutional Neural Network

A convolutional layer is basically a set of learnable filters (kernels) which represents a specific part of the image by preserving the spatial relationship between pixels. It is activated when it detects some specific type of feature at some spatial position of the input. The pooling layer (subsampling) reduces the dimensionality of each feature map but retains the most important information. This is done by a few common methods; max, average, sum pooling. The result is a smaller and more manageable feature dimension, with lesser number of parameters and computations. The final fully connected layer involves a softmax function which will help us make the prediction, by exponentiation and then normalizing the inputs.

$$\boldsymbol{softmax}(\boldsymbol{x})_i = \frac{\boldsymbol{exp}(x_i)}{\Sigma_j \boldsymbol{exp}(x_i)}$$

The output of the softmax function is used to represent the categorical distribution that gives us a list of values from 0 to 1 that add up to 1, which represent the probabilities (confidence) of each class prediction.

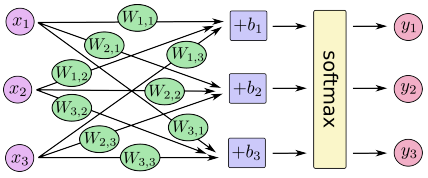
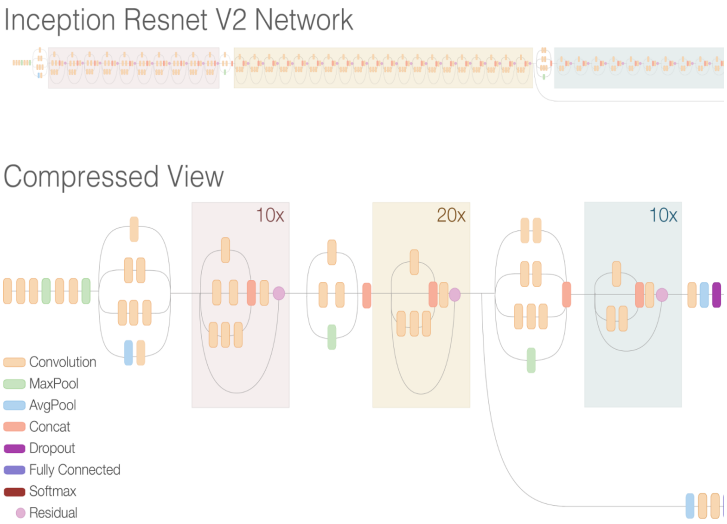


Fig 2. Visualization of softmax function implementation in the final layer of CNN models.

3.1 Inception-ResNet-V2



Inception-ResNet-v2 is a convolutional neural network (CNN) that achieves a new state of the art in terms of accuracy on the ILSVRC image classification benchmark as shown in the table below. Inception-ResNet-v2 is a variation of earlier Inception V3 model which borrows some ideas from Microsoft's ResNet papers.

Model	Version	Acc@1	Acc@5
InceptionResNetV2	TensorFlow	80.4	95.3
InceptionV4	TensorFlow	80.2	95.3
ResNet152	PyTorch	78.428	94.110
InceptionV3	PyTorch	77.294	93.454
ResNet101	PyTorch	77.152	93.548

4 Datasets

Since the application is geared towards monitoring surveillance video and detect threat perception and theft scenarios, as one of the areas of focus, it seemed natural to choose datasets containing images of handguns, knives, human hand and everyday objects observed in retail environment. Using these collection of images, I prepared three class of image datasets. a) guns b) knives c) hand and d) Everyday Objects observed in retail environments, and created over 1000 labels accordingly in the database. Images from following data sources were used for this purpose.

Knives Images Database, which contains 9340 negative examples and 3559 positive examples, **Internet Movie Firearms Database**, which contains 8557 images, **Hand Dataset** which contains about 14700 hand images from various sources. **EgoHands** Dataset containing 120000 images. **ImageNet** dataset. which contains more than 1.2 million images in over 1000 categories.

5 Observations and Results

I trained and evaluated the data set on different models to see which one gives the best result

5.1 AlexNet and Caffe

I started the process by using AlexNet in combination with Caffe. I provided the training model, a dataset containing three classes of image; handgun, knives and human hand as an input and it took about 30 mins for the training process to complete. I divided the dataset into training and validation in ratio of 90% - 10% respectively. Following were the training parameters used at high level.

Training Epochs	Solver Type	Base Learning Rate
30	Stochastic Gradient Descent	0.01

As seen from the graph in Fig 3, the model managed to reach ~90% validation accuracy after 30 Epochs.

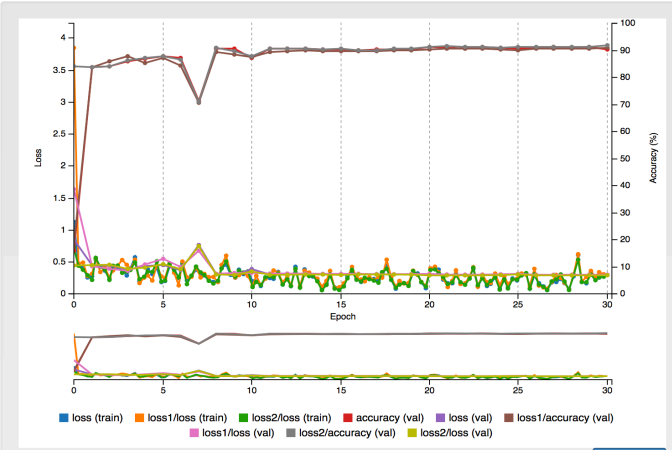


Fig 3. Accuracy and Loss Graph of training and validation steps

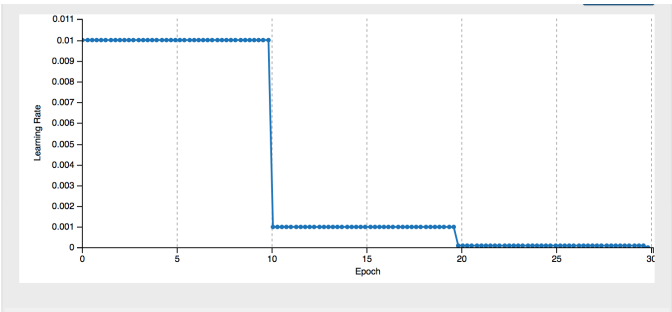
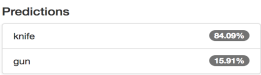
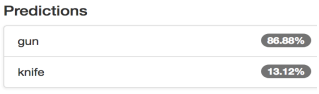
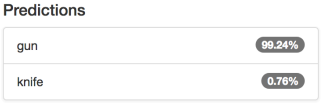


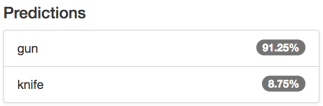
Fig 4. Learning Rate v/s Epoch for AlexNet/ Caffe

The Figure 5. Below shows a sample of the test performed on some images to see the accuracy of the model. As seen the Model is not reliable in predicting images of class knife. It predicted images of gun class reliably for some of the test images.









5.2 GoogleNet and TensorFlow

As before I divided the dataset into training and validation in ratio of 90% - 10% respectively. It took approx. 30 mins for it to train on the dataset. Following were the training parameters used.

Training Epochs	Solver Type	Base Learning Rate	Gamma	Step Size
80	Stochastic Gradient Descent	0.001	0.96	10

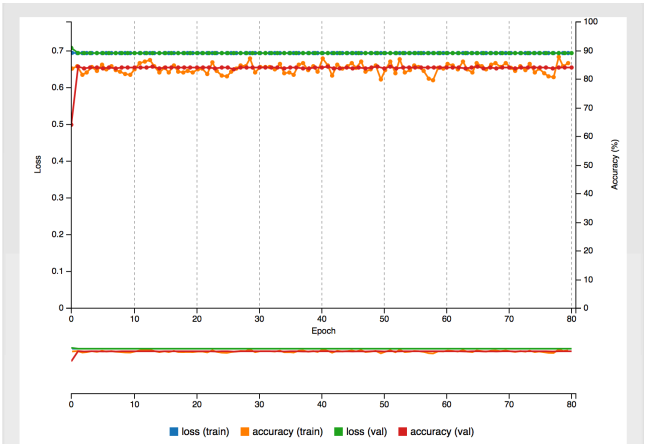


Fig. 6 Accuracy and Loss Graph of training and validation steps

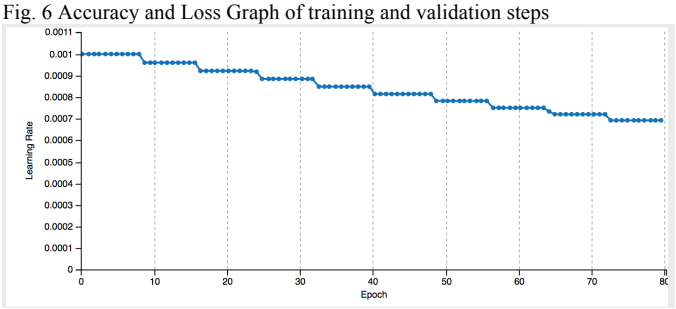
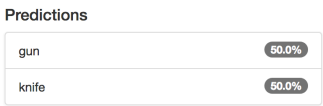
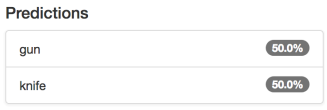


Fig. 7 Learning Rate v/s Epoch for GoogleNet & TensorFlow

The model achieves ~85% validation accuracy afer 80 Epochs.

Fig 8. Below shows result of some the Test Image used against this model.





The model is not accurate in predicting the image class as seen from above tests.

5.3 Inception-ResNet-V2 and TensorFlow

The dataset was divided into training and validation in ratio of 90% - 10% respectively. It took approx. 10 hrs for it to train on the dataset. Following were the training parameters used.

Training Epochs	Solver Type	Base Learning Rate
100	Stochastic Gradient Descent	0.0001

As can be seen from the graphs below, the model achieves 99.97% accuracy after 100 Epochs.

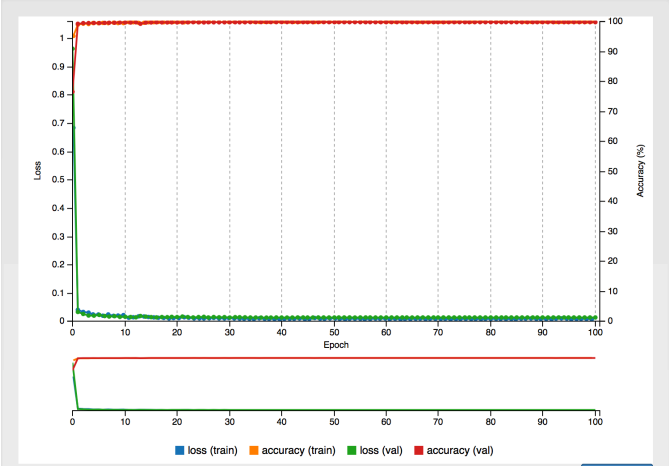


Fig. 9 Accuracy and Loss Graph of training and validation for Inception-ResNet-v2/TensorFlow

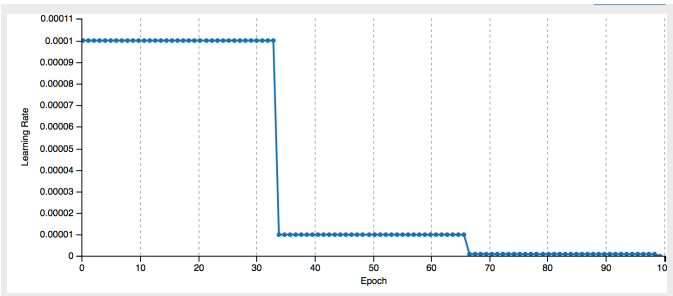


Fig. 10 Learning Rate v/s Epoch for Inception-ResNet-v2 & TensorFlow

Below figures show some results of Testing some images on the Model.

Predictions

gun	57.33%
hand	32.69%
knife	9.98%

Predictions

knife	98.42%
gun	1.58%
hand	0.0%

Predictions

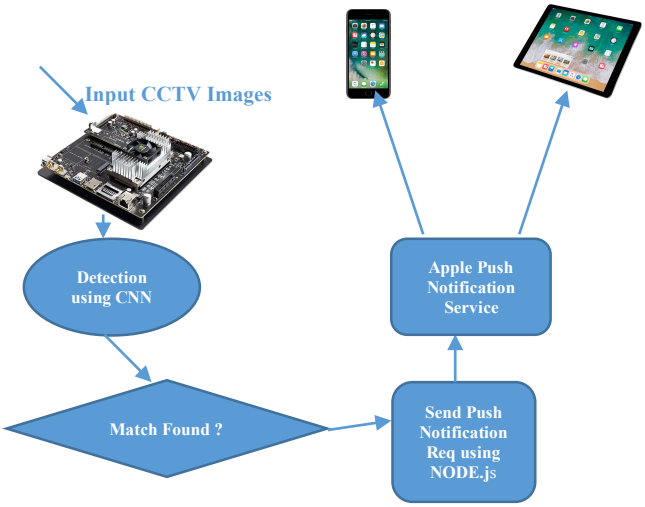
gun	88.41%
knife	8.69%
hand	2.9%

As can be seen from above results, the model is fairly accurate in predicting the image class.

It is to be noted that all the above training was carried out on a Intel i7 12 Core CPU based system with 64G RAM, and having NVIDIA M6000 GPU.

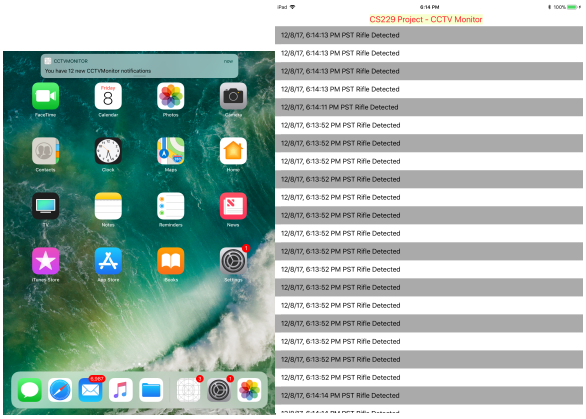
5.4 End To End Solution Running on Target Platform

The figure below shows the complete end to end solution in terms of system setup and the application workflow. The trained network (Inception-ResNet-v2) was deployed on the target platform NVIDIA Jetson TX2, and was applied on each and every image captured by the onboard CCTV Camera module in real-time, to monitor Images Of Interest (I-O-I).



5.5 iOS App Receiving Notifications

The figures below show iOS app receiving real time notifications from the Nvidia Jetson TX2 board, detecting images of interest (IOI).



6 CONCLUSION

Based on the observed results, Inception-ResNet-v2 used with TensorFlow Framework is giving the best accuracy and results out of the models evaluated and analyzed, and for the most part it detects the Images Of Interest accurately when deployed on the target platform.

7 FUTURE WORK

Despite Inception-ResNetV2 performing the best, I found that many predictions had a probability of 20% to 40%, even if these predictions were correct. The first step I would like to take is to increase the confidence in these predictions so that the model would be more well trained. This could be done my training it on more data or increasing the epochs when training the CNN. Also after developing an end-to-end Proof Of Concept solution, I strongly feel that it has the potential of becoming a commercially viable product

8 REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105, 2012.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” arXiv preprint arXiv:1408.5093, 2014.

[3] Convolutional Neural Networks (CNNs / ConvNets), http://cs231n.github.io/convolutional-networks/

[4] Scalable Object Detection using Deep Neural Networks Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov Google, Inc.

[5] Automatic Handgun Detection Alarm in Videos Using Deep Learning
Roberto Olmos, Siham Tabik, and Francisco Herrera
Soft Computing and Intelligent Information Systems research group Department of Computer Science and Artificial Intelligence,

[6] CCTV object detection with fuzzy classification and image enhancement, Andrzej MATIOLAŃSKI, Aleksandra MAKSIMOWA, Andrzej DZIECH, Multimedia Tools and Applications, 2015

[7] Automated Detection of Firearms and Knives in a CCTV Image, Michał Grega, Andrzej MATIOLAŃSKI, Piotr Guzik, Mikołaj Leszczuk, Sensors, ISSN 1424-8220

[8] TensorFlow, An open-source software library for Machine Intelligence. https://www.tensorflow.org/

[9] Rethinking the Inception Architecture for Computer Vision
Christian Szegedy Google Inc. szegedy@google.com
Vincent Vanhoucke vanhoucke@google.com,
Sergey Ioffe sioffe@google.com
Zbigniew Wojna University College London
zbigniewwojna@gmail.com