Robo-Nanny: ConvNets for Intelligent Baby Monitoring

Lily Cheng - June 2017
Stanford University - CS231n Final Project Report
lilcheng@stanford.edu

Abstract

In this project, we seek to apply supervised learning techniques on state-of-the-art deep convolutional neural networks (CNN) to infer a baby's status inside a crib using a baby monitor video feed. A dataset of 2500 images assigned to 5 predefined classes was used for training and hyperparameter tuning. After comparing different model architectures including ResNet18[1], AlexNet[10] and SqueezeNet[6], the best class-weighted accuracy of 96.7% on test set was achieved on pre-trained ResNet18. Despite being significantly smaller model, Squeezenet was able to deliver a test accuracy of 94.6% and, in a real life application, would be preferable due to efficient model size. When tested using images captured using a different camera position or found online, the model was found to have poor domain adaptability and further work would be need to done to improve the ability of the model to accurately classify images drawn from unseen domains.

Figure 1. Sample images from four classes

1. Introduction

Parents often utilize baby monitoring devices to keep an eye on young children during the night. However, the alert mechanism of such monitors are typically triggered on sound and not visual information. Not all cries at night require adult intervention. Child monitoring devices would have greater utility if they are able to consider both visual and audio information to determine whether an alert needs to be triggered. This would result in fewer unnecessary alerts and hence better sleep for the parents. For example, if the baby is crying but still lying down, chances are no intervention is needed to drift back into sleep.

The objective of this project is to apply state-of-the-art CNN to automate the identification of a childs status from a visual monitoring device. In addition to optimizing the network for high accuracy, a comparison across different architectures will be provided to understand trade-offs between accuracy and compute/memory requirements of a forward pass, recognizing the finite compute and memory available in low-cost consumer hand-held devices.

2. The Problem

Working with limited and unbalanced data. While CNNs are known to be able to achieve high accuracy rates if given enough data for supervised image classification, in practice, the resources needed to collect large amounts of videos and images of children could be a hurdle for commercialization. It is resource intensive to find a large number of parents who will offer video streams of their child in the crib given the privacy concerns. For this project, 2500 images were been collected for one baby in one crib/camera configuration over two weeks. It is a core objective of this project understand what levels of accuracy can be achieved with such a limited dataset by leveraging transfer learning and fine-tuning of pre-trained models.
In addition, due to the fact that the baby is mostly lying down in a crib, the dataset is highly unbalanced and techniques such as oversampling or weighted-class loss functions would be needed to avoid classification bias.

Extracting salient frames from video stream. Given that the video feed is of a sleeping baby, most of the frame-to-frame pixel changes at night would be minimal with the exception of when the child is moving. It is proposed that motion detection techniques such as image subtraction [2, 16] or optical flow [9] be used to identify and extract key frames from the video stream for use in training. In practice, the child monitoring device would likely only invoke the inference function if it detected some change event such as a sound or movement, hence this method is reflective of a real-world application.

Model architecture selection and optimization. To understand the maximum achievable accuracy, different model architectures such as Alexnet[10] and ResNet[1] will be tested using this dataset. To further understand the impact of transfer learning, pre-trained model weights will be used with different degrees of fine-tuning by freezing/unfreezing different layers of weights during training.

Further more, as an edge computing application where the inference step will be processed on a mobile device with memory, computation and power consumption limitations, this project seeks to also understand and explore state-of-the-art architectures such as SqueezeNet[6]. The trade-offs between accuracy and resource requirements will be quantified through testing.

Understanding and overcoming domain bias. In practice, a commercially viable application would need to be able to accurately classify images drawn from unseen target domains. For example, there can be significant domain shift caused by variations such as different camera viewpoints, different rooms/cribs, different babies, different lighting conditions and different toys inside the crib.

This project seeks to understand the domain adaptability of the trained model by testing the model against datasets with some domain shift introduced or collected using a completely different method. In addition to training the model with random image cropping, flipping and rotation augmentation techniques[3], it is a hypothesis that cropping the unseen datasets using object localization could reduce the sensitivity of the model to camera placement and zoom level. These techniques were tested to understand potential impact on improving domain adaptability of the model.

Network Visualization. CNN networks have often been criticized for being a black box. Network visualization techniques will be used on the train model in an attempt to gain some intuition for what the model is looking for. Given that each class is actually a collection of the baby in many different positions in the crib, would we get more recognizable images if we trained pixels to maximize a certain neuron activation in an earlier layer rather than the final output layer?

3. Related Work

Addressing unbalanced datasets. The problem of class imbalance is well documented and is known to lead to classifiers that are biased towards major classes with very poor classification rates on minor classes. In the extreme case, it is possible for classifiers to class everything as the major class. To avoid such problems, Longadge et al.[11] outlined solutions that fall into three basic categories: algorithmic, data-preprocessing and feature selection. Existing literature suggest that data pre-processing provides the best solution by adding or subtracting data through over and under-sampling to balance the underlying data. However, in this project, there are significant constraints to obtaining more samples for under-represented classes, hence, the algorithmic approach would be appropriate.

Motion Detection. Some of the most common methods for human motion detection used in video surveillance are outlined by Alzughaibi et al. [16]: the temporal difference method, optical flow method and background subtraction method. The temporal difference method is one of the easiest models and involves taking the pixel-wise difference between the previous frame and the current frame and comparing it against a threshold. The optical flow method uses an algorithm which produces a 2D vector field representing the displacement vector of each pixel between two frames[9]. The background subtraction method is similar to the temporal difference method in that the pixel-wise difference of the current image is subtracted from a defined background image. The background image can be initialized using various methods, including taking the median of a series of frames known to contain only the background. Such a background can be updated to adapt to changes over time. In our application, it is likely that a temporal difference method (also referred to as image subtraction method) will suffice though we also look to test the optical flow method for the sake of comparison.

CNN Architectures. A paper comparing different deep neural networks for practical applications by Canziani et al. published in April 2017[17] provides a comprehensive framework for model selection based on not only accuracy but practical considerations towards memory footprint, parameters, operations count, inference time and power consumption. Top performing network families in the last four years on the ImageNet challenge such as Alexnet[10], NIN, ENet, GoogLeNet, VGG, ResNet[1] and Inception models were included. In addition, Squeezenet [6] was introduced by Iandola et Al. in Nov 2016 as an alternative to AlexNet which achieves the same level of accuracy whilst having 50x fewer parameters than Alexnet. Also worth noting is when compression techniques proposed by Han et al. [14] such as network pruning and quantiza-

tion are applied to SqueezeNet, 510x size reduction was achieved compared to uncompressed AlexNet without any degradation in accuracy. Compressed SqueezeNet requires less than 0.5MB of memory. While a comprehensive survey of all of these models is out of scope for this project, a few selected architectures will be compared to gain an intuition for the accuracy and resource utilization trade-offs for this specific dataset.

Domain Adaption. In response to the quest to come up with the next competition-winning model, Torralba and Efros [13] noted that the community has focused on accuracy at the expense of cross-dataset generalization capability of winning models. If these models are to be representative of the real world, they should be able to be trained on ImageNet, say, and then accurately classify images from datasets such as PASCAL VOC, SUN09 and so on. To a human, all of these datasets are from the same "domain" - the world. However, in practice, these public datasets inevitably suffer issues such as selection bias and capture bias, reducing their ability to truly represent the real world. In this project, it is likely that the our dataset suffers from significant bias. Although a full solution may not be viably found within the scope of this project, it is important to quantify and acknowledge the limited ability of this model to achieve the kind of cross-domain adaptability that would be needed in real-world applications.
A comprehensive survey of techniques to allow a model trained on a source domain to classify an unseen dataset drawn from a different domain was presented by Csurka[8]. One of the more promising methods are deep domain adaptation techniques using deep networks. For example, the concept of Generative Adversarial Networks can be applied in the domain adaptation context[15]. The discriminative model seeks to accurately discriminate between images from the source-domain and the target-domain. The generative model aims to generate target-domain images that appear to have been drawn from the same distribution as the source-domain, hence bridging the gap between the two domains.

Network Visualization. In attempts to gain more intuition about what deep neural networks are doing to achieve the stated objective, a number of techniques have been developed over the years to visualize the trained networks. In particular, a method presented by Simonyan et al. [12] involves generating an image by gradient accent with respect to the image pixels, treating the weights as fixed parameters, with the optimization objective of maximizing the selected class score. This technique will be applied in this project in an attempt to visualize different layers of the network.

4. Data Collection and Preprocessing

4.1. Main Dataset - A1 and A2

For this main dataset, an IP camera in a fixed position relative to a baby crib was used to capture videos of a single baby. Dataset A1 and A2 are similar but A2 has 25% more samples . The following 7 pre-processing steps were used to produce the input to our CNN model for training, validation and testing.

Step 1: Capture Video Stream. A Nest IP Camera was placed over a baby crib and a continuous stream of frames were captured at a rate of 1 frame per 10 seconds. This script was activated every time the child is put into the crib for sleep over a period of 2 weeks.

Step 2: Extract Salient Images from Video. Two techniques were briefly tested in order to efficiently extract salient images from the video feed: image subtraction[2, 4] vs. Optical Flow[9] The image subtraction method was selected due to the combination of efficacy and simplicity. The salient images extracted represent 5% of the total raw frames captured.

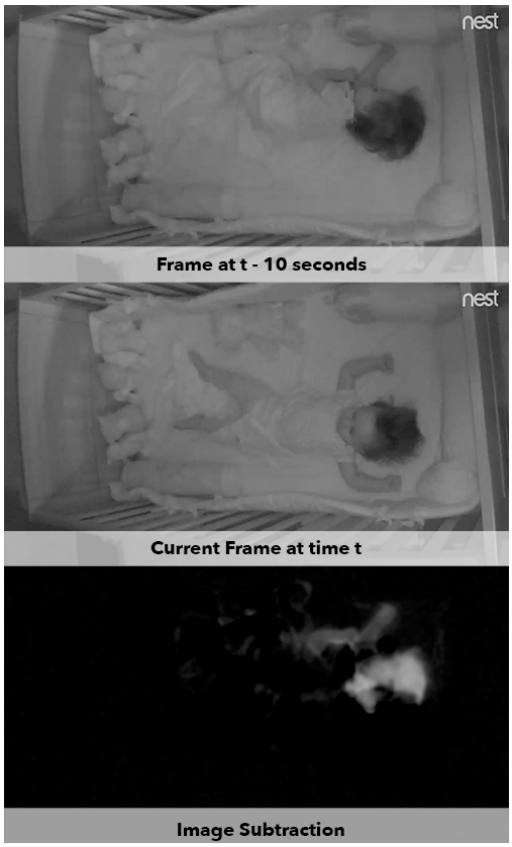

Figure 2. Image subtraction operation.

$$\tau_{lower} < |P[F(t-1)] - P[F(t)]| < \tau_{upper}$$

where pixel values P of a frame F extracted at time t is subtracted from the frame immediately preceding it to obtain the pixel difference. For noise reduction, a Gaussian filter was applied to the frames prior to image subtraction. If the L1-norm is within a lower and upper threshold, the image is extracted for use. The upper threshold helps to eliminate frames where camera jitter or illumination changes caused a high pixel difference rather than localized motion.

Step 3: Manually classify into 5 classes. The resulting images were manually classified into the following 5 categories, providing a class distribution as shown in the table shown below.

	Number of Images By Class			
Class	A1	A1%	A2	A2%
Caretaker	40	2%	51	2%
Empty Crib	182	9%	278	11%
Sitting Up	209	10%	269	11%
Lying Down	1165	58%	1462	58%
Standing Up	406	20%	440	18%
Total	2002	100%	2500	100%

Table 1. Dataset sample count and class distribution prior to augmentation.

Step 4: Split into Train, Validation and Test Sets. This dataset was further split at random into training, validation and test set at a ratio of 68:20:12.

Step 5: Resize and Crop images. The resulting images were converted into squares of 224x224 pixels (or 227x227 depending on the model used) by first scaling the shorter dimension to be 224 pixels and cropping the center of the image along the longer dimension.

Step 6: Synthetic augmentation (selectively applied). This step was only introduced later on in the training to produce dataset A2* (which is an augmented version of dataset A2) so that the effect of augmentation can be evaluated. In this step, 3 additional images were synthetically generated from each training image by flipping along vertical and horizontal and both axes.

Step 7: Normalization. The images were then normalized using the mean and standard deviation specified by the pre-trained model.

4.2. Dataset B - modified camera viewpoint

In this dataset B, the same set up as the main dataset A was used with the exception that the camera was more zoomed out. In the two images at the top of figure 3, one can see that the camera is capturing more of the room and not just the scene inside the crib as per dataset A.

The purpose of this dataset is to test the adaptability of the model to changes in camera viewpoint and to further explore whether adaptability can be improved by localizing the object and automatically cropping prior to feeding into the network for inference. Steps 1,2,3,4 and 7 outlined in section 4.1 was used to process this dataset. Step 2 was modified so not only was image subtraction used to identify the salient frames, it was also used to localize the object (shown in figure 3 - outlined by box labeled "A"). The localized zone was then programmatically expanded to the larger box "B" to ensure that the cropped image captures most of the baby and not just the moving body parts. Dataset B includes a total of 68 images across the 5 classes.

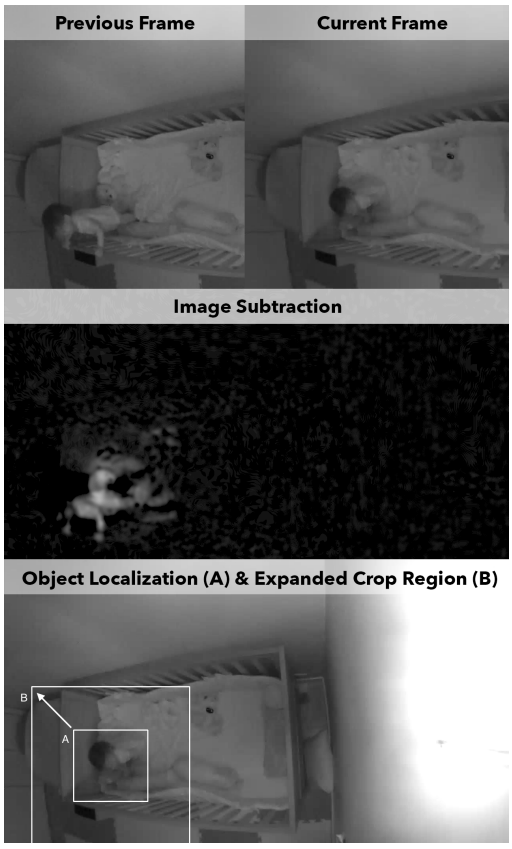

Figure 3. Object localization and automatic cropping used to reduce viewpoint sensitivity.

4.3. Dataset C from Youtube/Google

In this third test dataset, a small collection of 155 image samples of different babies in different positions inside the crib has been collected from images and videos on Google and Youtube. Images that had a top-down angle rather than taken from side of the crib where selected.

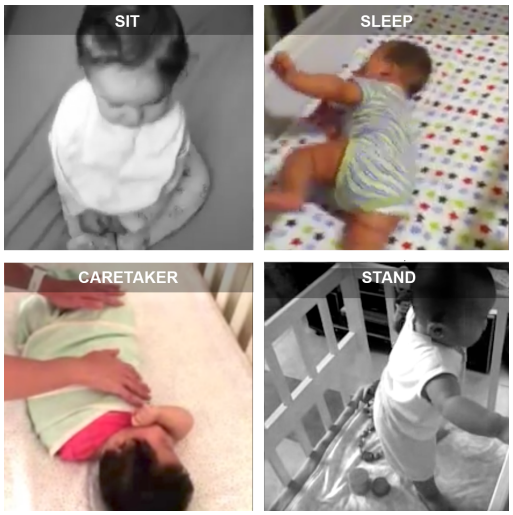

Figure 4. Images of babies in cribs from Youtube/Google.

5. Methods and Algorithms

5.1. Model Architectures and Transfer Learning.

AlexNet, ResNet18 and SqueezeNet pre-trained on ImageNet were tested. In order to adapt the first two models to produce class scores required for this dataset, the final fully connected layer was modified to provide 5 outputs scores. SqueezeNet does not have any fully connected layers, hence the final convolutional layer in the model classifer was modified to produce the required output. In addition, in order to compare the difference between using the pre-trained weights as-is compared to simply using the weights as an initialization value, our ResNet18 model was used in two ways: Firstly with weights frozen with other than the last layer for fine-tuning (referred to as "ResNet18-Fr" in Table 3); Secondly with all weights unfrozen.

A random search was used to tune the model hyperparameters using the validation dataset. Finally, the test dataset was used to quantify model performance.

The table below outlines the relative resource requirements of the model architectures tested, including number of parameters, number of operations and memory requirements - base model and with deep compression applied. One can see a significant practical advantage to using SqueezeNet so to the extent that accuracy can be demonstrated to not be significantly impaired.

Model	Param	GFlop	Memory
ResNet18	11M	3.5	44.6MB $\rightarrow$ 4MB
AlexNet	60M	1.5	250MB $\rightarrow$ 6.9MB
SqueezeNet	1.25M	0.72	4.8MB $\rightarrow$ 0.47MB

Table 2. Resource requirements of different architectures.[24, 18]

5.2. Optimization Algorithm and Loss Function.

Stochastic gradient descent with momentum was used as the optimizer for all tests.

$$v_{t+1} = \rho * v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha * v_{t+1}$$

where ρ refers to the "friction", v refers to velocity and x refers to weight values. In terms of the loss function, the objective of the optimization is to maximize prediction accuracy on test set with equal importance assigned to every class, hence a weighted cross-entropy loss function [5] will be used to address the unbalanced nature of the dataset.

$$Loss(X, C) = W[C] * (-X[C] + log(\sum_j exp(X[j])))$$

where X are the images, C are the classes and W is the weights applied to each class to balance the data.

5.3. Data Visualization.

In order to visualize what the model might interpret to be an image that belongs to a given class, a gradient ascent method[12] with respect to randomly initialized input image pixels using fixed weights of a trained Alexnet model was used to maximize the output score for the "stand" class. Given that the baby could've been standing in many different locations in the crib, it is expected that perhaps the resulting image would look like a composition of many instances of the baby in different positions across the image and, hence, difficult to see. We hypothesized that if the gradient ascent was replicated to maximize the activation of a neuron in the second last layer, with the neuron being chosen to correspond to one that is multiplied by high weights to produce the "stand" class score, perhaps the image generated would be more interpretable by humans (for e.g. representing one instance of the baby standing in one spot).

$$I^* = argmax N_y(I) - \lambda \|\mathbf{I}\|_2^2$$
$$where, N_y = argmax(W_{FC2}[y, :])$$

The method was extended further to the third layer.

6. Results and Conclusions

6.1. Motion Detection Method

As previously noted, both image subtraction and optical flow were explored briefly as possible methods for motion detection. Qualitative results are presented in a Youtube video available at the following link: https://youtu.be/sF24ccg9-Cs

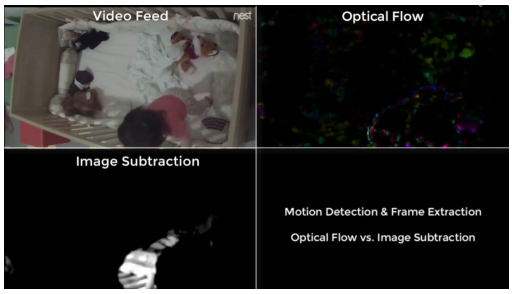

Figure 5. Youtube video comparing image subtraction vs. Optical Flow alongside original video.

Optical flow did not work as well as image subtraction during the testing on first attempt. There was significant noise around the bars of the crib and the actual baby motions were less noticeable with only the outlines being shown as active vectors. Perhaps optical flow is not so suitable for this task as the frames captured were 10 seconds apart so in order to capture such large displacements in between frames, a coarser scale would've had to be used at the expense of accuracy. Although optical flow could've been optimized for the task at hand, image subtraction was sufficient and had the benefit of being simple and easy to execute, hence, image subtraction was used in our pre-processing steps.

6.2. Model Architectures vs. accurate rates

The best result was achieved using a pre-train ResNet18 finetuned with unfrozen weights using the A2 dataset with synthetic augmentation. Weighted training accuracy of 98.2%, a weighted validation accuracy of 94.7% and a weighted test accuracy of 96.7% were achieved.

It should also be noted that ResNet18 performed significantly better with larger A2 dataset compared to A1 dataset. The synthetic augmentation step preformed on the A2 dataset (noted as A2*) also exhibited a small minor improvement in test accuracy results. Resnet18 with frozen weights with the exception of the final layer performed the worse with test accuracy of 81.2%. Given that this dataset is significantly different from Imagenet (on which the models were pre-trained on), simply finetuning the weights of the final layer was apparently not sufficient to fully optimize the model to the new dataset.

Alexnet and Squeezenet performed at similar levels, both being marginally worse than ResNet18. However, as noted in Table 2, SqueezeNet was designed to be extremely efficient with only 1.25M parameters, 0.72 GFlops per image and a compressed size of 0.47MB. This has significant practical benefits including power consumption and low latency. Given these accuracy results, it appears that Squeezenet presents the best trade-off between accuracy and resource requirements.

		Weighted Accuracy		
Network	Data	Train %	Val %	Test %
ResNet18-Fr	A1	86.9%	85.7%	81.2%
ResNet18	A1	100%	93.5%	90.5%
ResNet18	A2	99.9%	94.2%	96.0%
ResNet18	A2*	98.2%	94.7%	96.7%
AlexNet	A1	97.5%	93.7%	94.2%
AlexNet	A2*	97.5%	94.3%	95.0%
SqueezeNet	A2*	99.3%	92.5%	94.6%

Table 3. Comparing results using different model architectures and datasets.

	Ground Truth				
Pred	Care	Empty	Sit	Sleep	Stand
Care	5	0	0	0	0
Empty	0	31	0	0	1
Sit	0	0	24	2	0
Sleep	0	0	0	152	1
Stand	1	0	0	0	52
Total	6	31	24	154	54
Recall	83%	100%	100%	99%	96%

Table 4. Confusion matrix of ResNet18 on A2* test set

6.3. Results of Inference Dataset B

The test results of dataset B using models trained on dataset A2/A* were poor but there are promising avenues for further improvement. Recall that dataset B was captured using the same setting as Dataset A other than the camera being configured to capture a wider scene (more zoomed out). In the table below, "Not cropped" refers to dataset B, being used as-is whereas the "cropped" set utilized the image subtraction object localization technique to automatically crop the salient part of the image for further processing.

		Test Accuracy on Dataset B	
Model	Train Dataset	Not Cropped	Cropped
AlexNet	A2	23.8%	40.3%
AlexNet	A2*	35.5%	51.8%
SqueezeNet	A2*	40.3%	54.4%

Table 5. Cross-domain adaptability of trained models.

With 5 classes, random guesses would result in accuracy rates of 20%, hence the Alexnet trained on A2, tested on uncropped dataset B is not much better than random. However, we do observe that the image localization and cropping significantly improved the results from 35.5% on Alexnet to 51.8% . In addition, we can see that the synthetic data augmentation performed on dataset A2 helped to improve the model adaptability to this unseen dataset. AlexNet trained on A2* (augmented) exhibited a 51.8% accuracy rate on the cropped dataset B compared to 40.3% when A2 was not augmented. Further more, it appears that using the same A2* dataset as training data and cropped dataset B as test data, SqueezeNet was able to achieve a 54.4% accuracy, somewhat better than AlexNet with the same set up. Further exploration would be required to fully understand this effect.

Although the best result obtained here with Squeezenet at 54.4% is nowhere near the level required for commercialization, it should be noted that the techniques employed here are relatively simple and have already shown marked improvements. Hence, it can be concluded that there may be other methods that can further improve the cross-domain adaptability of the model to unseen data.

6.4. Results of Inference Dataset C

Recall that Dataset C was obtained from Youtube and Google of completely different babies, in different cribs, camera settings and light settings. Although from the perspective of a human, these are all still pictures of babies in cribs and hence would be desirable to be considered as one domain, from the perspective of the machine, it is a significantly different domain.

When tested on AlexNet A2 dataset, the weighted classification accuracy on dataset C was not much better than random at 23.5%. When the model trained on the augmented A2* dataset was used, the result improved to 30.4%.

Although the accuracies on dataset A achieved were in the high nineties, it is sobering to realize that the model really doesn't generalize well at all to unseen images that humans would consider to be from the same "domain" - have we created yet another "Clever Hans"? The model is overfitting to the domain represented by dataset A and optimizing for the idiosyncracies manifested in that dataset in such a way that it no longer is able to perform the task for "similar" images.

To achieve a level of cross-domain adaptability required for commercialization, further work would need to be done to explore more sophisticated Domain Adaption methods such as those mentioned by Csurka[7] as well as the obvious approach of broadening the source domain by capturing more data so it is more representative of the target domain.

6.5. Results of Visualizations

Below are two images generated through gradient ascent on pixel values for the class "sit" and the class "stand" respectively. A trained AlexNet was used as the base model. It can be seen in the visualization for the "sit" class that something resembling the top of a baby's head can be seen in various positions across the image. Likewise, in the visualization for "stand", one can see artifacts towards the bottom of the image which looks like arms holding on to the edge of the crib.

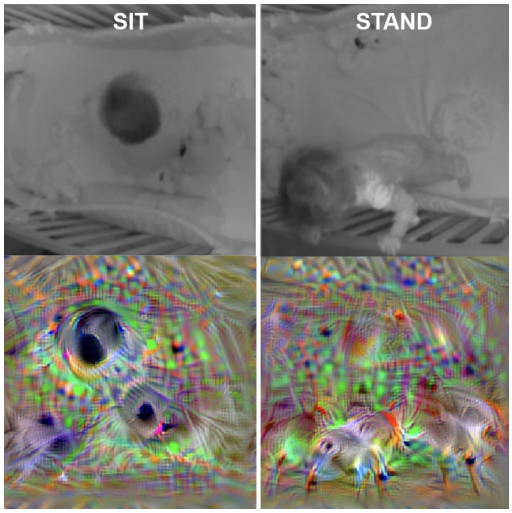

Figure 6. Network visualizations of two classes: sit and stand.

Below are the visualization attempts to visualize the 2nd last and 3rd last layers of the network by selecting neurons that correspond with high weights for the output class.

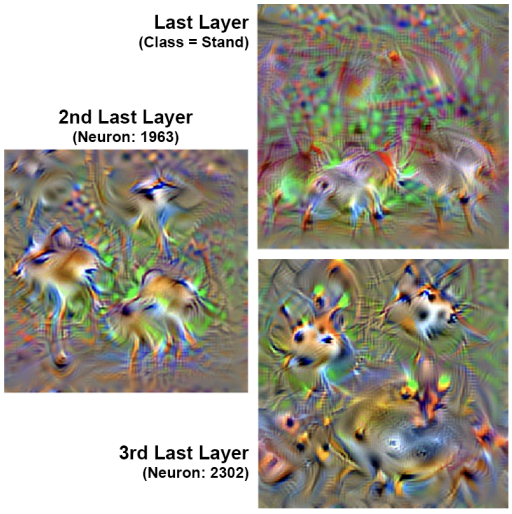

Figure 7. Network Visualizations of 2nd and 3rd last layer.

Although the 2nd/3rd last layer visualizations do not val-

idate the hypothesis that they would be decompositions of the final layer class image, it is interesting to see that they are somewhat more interpretable than the class image. For example, the 3rd last layer one looks like fox faces (a stuffed toy present in the crib).

Finally, it is also interesting to see the differences between class visualizations generated by ResNet18 shown below compared to the previous ones from AlexNet. ResNet images are significantly more intricate - more work would be required to understand how to account for these visualization differences. Similar to the class image for "sit" for AlexNet, one can see an oval that holds some resemblance to the top of a baby's head whilst sitting down.

Figure 8. Visualization of class = Sit generated from ResNet18.

7. For Further Investigation

Although a respectable 96.7% test accuracy was achieved, the dataset collected was too limited and not truly representative of the real world domain. In order to improve the real world applicability of the model, there are four primary avenues that can be pursued:

1. More data should be collected which include different camera viewpoints (zoom level and angle), different babies in different settings. Ideally, the data collected should have the same degree of variation as what one might encounter in real life - such as different color of cribs, babies of different ages, sizes and color, different bedding and so on.

2. Given the challenges of collecting more data, additional methods of synthetic image augmentation should be explored. When dataset A2 was synthetically augmented by flipping across vertical/horizontal axes, the incremental benefit on test accuracy on the same domain was limited but the benefits on the cross-domain adaptability of the model was significantly increased. Perhaps color and brightness adjustments can be considered to synthesize lighting scenarios.

3. It would be challenging to collect a completely representative dataset for training that is free from any kind of bias, hence it would be good to further investigate and test the Domain Adaptation techniques outlined in the Csurka paper[8]. For example, the technique we explored with Dataset B with automated localized cropping was able to make Dataset B more similar to what the model has seen before in dataset A and was able to deliver a marked improvement in accuracy. In particular, generative adversarial methods that can modify new target domain images to the source domain seems interesting to explore - though that method may be too resource intensive and introduce too much latency to deploy in real-time.

4. Lastly, of course, changes or constraints can be designed into the application to make the image recognition problem easier and more reliable to solve. For example, depth sensors are becoming more prevalent in mobile devices and could bring the additional information that a model needs to more reliably understand what is the going on in the environment. Also, some commercially available intelligent baby monitors (such as the Nanit Baby Monitor) provide a physical fixture to be attached to the side of the crib, so to minimize the camera viewpoint variability in different settings. Finally, some applications may be able to guide the end user to provide some labeled training data from the new domain so the model can be finetuned for each individual user scenario. For example, when the Touch ID feature of the iPhone is activated for the first time, the user is guided to provide a number of different finger-prints scans around different parts of the finger, so the device can recognize the print even if not placed exactly in the same way on the sensor every time.

Finally, a more detailed and comprehensive architecture test can be conducted, including applying deep compression [14, 24, 18] to the model to confirm that the best architecture is indeed a compressed Squeezenet.

8. Code References

- Code libraries used: Pytorch[19], OpenCV[20]
- Third party code referenced: Motion Detection with OpenCV[21], Transfer Learning in Pytorch[22], Network Visualization in CS231[23]
- Code used in this project can be accessed at Github repo: https://github.com/ririgriff/cs231n_project

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition.* 2015.

[2] Nayyab Naseem, Mehreen Sirshar *Target Tracking In Real Time Surveillance Cameras and Videos* 2015.

[3] Sebastien C. Wong, Adam Gatt, Victor Stamatescu, Mark D. McDonnell *Understanding data augmentation for classification: when to warp?* 2016.

[4] Massimo Piccardi *Background subtraction techniques: a review* 2004.

[5] Numerous contributors. *nn: Loss Functions (PyTorch Library*

[6] Forrest N. Iandola, Song Han, Mathew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer. *SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size*

[7] Sagie Benaim, Lior Wolf. *One-Sided Unsupervised Domain Mapping*

[8] Gabriela Csurka. *Domain Adaptation for Visual Applications: A Comprehensive Survey*

[9] Gunnar Farneback *Two-Grame motion Estimation Based on Polynomial Expansion*

[10] Alex Krizhevsky *One weird trick for parallelizing convolutional neural networks.*

[11] Rushi Longadge, Snehlata Dongre, Latesh Malik *Class Imbalance Problem in Data mining: Review*

[12] Karen Simonyan, Andrea Vedaldi, Andrew Zisserman *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

[13] Antonio Tarralba, Alexei Efros *Unbiased Look at Dataset Bias*

[14] S. Han, H. Mao, W. Dally *Deep compression: Compression DNNs with pruning, trained quantization and huffman coding.*

[15] I. Goodfellow, J Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio *Generative Adversarial nets.*

[16] Arwa Darwish Alzughaibi, Hanadi Admed Hakami *Review of Human Motion Detection based on background subtraction techniques.*

[17] Alfredo Canziani, Eugenio Culurciello, Adam Paszke *An Analysis of Deep Neural Network Models for Practical Applications.*

[18] J. Shen, V. Boddeti, N. Vesdapunt, K. Kitani *Learning Compressed Models for Pedestrial Detection.*

[19] Numerous Contributors. *Pytorch Library http://pytorch.org*

[20] Numerous Contributors. (version 3.2.0) *OpenCV Library http://opencv.org*

[21] Adrian Rosebrock. *Basic motion detection and tracking with Python and OpenCV http://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/*

[22] Sasank Chilamkurthy. *Transfer Learning Tutorial http://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html*

[23] Numerous Contributors. *CS231n Assignment 3 Network Visualization Code https://github.com/cs231n/cs231n.github.io*

[24] Song Han. *Efficient Methods and Hardware for Deep Learning http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf/*