An Efficient Convolutional Neural Network Computer Vision System to Prevent Sudden Infant Death Syndrome

Vivek S Bharati
Homestead High School, Cupertino, CA, USA
vbharati238@student.fuhsd.org

Abstract

Sudden Infant Death Syndrome (SIDS) causes infants under one year of age to die inexplicably. One of the most important external factors responsible for the syndrome, called an 'outside stressor', is the sleeping position of the baby. When the baby sleeps on the stomach with face down, the risk of SIDS occurring is very high. We propose a Convolutional Neural Network (CNN) based computer vision system that can alert caregivers on their mobile phones within a few seconds of the baby moving to a hazardous face-down sleeping position. The model processes real-time image feeds with a single efficient forward pass. It has a low computational load and a low memory footprint. This would allow it to be embedded in low power edge devices such as crib cameras. Processing at the edge would also alleviate privacy concerns in sending images into the network. The CNN architecture is composed of multiple sets of processing units, each unit containing a 2D convolutional layer with the Rectified Linear Unit activation function followed by a Max Pooling layer. The final layer in the architecture is a fully connected dense layer with the Sigmoid activation function and outputs three classes of sleeping position indicators. The seed corpus for the training dataset was generated from realistic baby dolls with diverse racial mix in three sleeping positions (face-up, turning, face-down). These seed images were used to generate additional images by applying various image transformations. We experimented with various numbers of convolutional processing units and dense layers as well as the number of convolutional kernels to arrive at the optimal production configuration. We

observed a consistently high accuracy of detection of sleeping position changes to turning and face-down positions with a trend towards even higher accuracies with caregiver feedback. Therefore, this system is a viable candidate for consideration as a non-intrusive technology to assist in preventing the Sudden Infant Death Syndrome.

1. Introduction

Sudden Infant Death Syndrome (SIDS) is a leading cause of death in infants under the age of one. Research has led the scientific community to believe that babies who succumb to SIDS are born with one or more conditions that may lead to a fatality in response to what are called internal and external stressors [1]. Researchers have identified three primary risk factors that must all be present and combine to result in the death of an infant from SIDS [2]. The first of the risk factors is the infant being vulnerable. Infants may be susceptible to an abnormality in the portion of the brain that controls heart rate and respiration. The second risk factor is the crucial nature of the developmental period of the baby. For example, during the first six months of a baby's life, rapid growth may destabilize the baby's internal systems. The third and most important risk factor is the set of outside stressors. These include things such as the sleeping position of the baby, especially when the baby sleeps on the stomach in a face-down position. All three risk factors listed above must be present for a SIDS fatality to occur. It is recognized by the experts that since the first two risk factors stated above cannot be seen or pinpointed, the most effective way to reduce the risk of SIDS is to reduce or remove outside stressors altogether [1,2]. This involves eliminating the sleeping position related risk, such as placing the baby on the back instead of on the stomach to sleep and ensuring the baby doesn't turn from a back sleeping position to a stomach sleeping position.

For young parents worried about the safety of a newborn baby, ensuring that the baby is in a back sleeping position when being put to sleep and that it stays in that position while sleeping is a key concern. Manually watching the baby either directly or via a baby monitor to ensure the avoidance of a sleeping position outside stressor is cumbersome and impractical. A number of tools have been created to attempt the task of avoiding the sleeping position outside stressor. These include baby sleeping sacks, breathing monitors, movement sensor pads, special mattresses, bumpers etc. The U.S Food and Drug Administration (FDA) has made it clear that a safe sleep environment for a baby is for it to be alone in the crib or bassinet, free of other objects including the devices mentioned above [4]. Therefore, it would be beneficial to create a non-intrusive and automated system that can be used to ensure the baby maintains a back sleeping position while asleep. The system should have no devices, objects or sensors in contact with the baby to prevent risks.

In this paper, we propose a convolutional neural network (CNN) based computer vision system that can process images from a camera that is focused on the baby's bed and can automatically detect changes to the sleeping position of the baby. The system will detect an attempt by the baby to change its position from the back sleep position to the stomach sleep position. The system will automatically generate a local audible alert and also send a notification to the caregivers' mobiles. The system also allows the caregivers to provide a validation feedback regarding whether the alert is a valid alert or is a false positive. This feedback is used to retrain the system so that the accuracy levels are increased. The system requires low computational power and a low memory footprint, which allows it to be embedded in edge devices such as crib cameras and alleviate any potential privacy concerns involved in sending images of the sleeping baby into the network for processing.

We have pre-trained the model using a seed data set created by using realistic baby dolls of multiple racial backgrounds and applying image transformations to obtain additional training images. Our tests on the trained model indicates a near perfect detection of the sleeping position outside stressor and a change of sleeping position from the back sleeping position to the stomach sleeping position.

Fig. 1. Sleeping position classifications

2. Related Work

The use of Convolutional Neural Network (CNN) systems for various computer vision tasks is well established, inspired by the work of Hubel and Wisel [5] on the vision systems of animals. LeCun et al [6] pioneered the use of CNNs for image classification tasks with their LeNet proposal. Krizhevsky et al [7] used more complex multi-layer CNNs along with techniques such as dropout and normalization to solve large-scale image recognition challenges. They also set the stage for hardware acceleration using GPUs that decreased the training time of large-scale models. Very deep convolutional networks for large-scale image recognition were proposed by Zeiler et al [8] along with pre-training and weight initialization as ways to solve the vanishing gradients problem in large convolutional networks. GoogLeNet proposed by Szegedy et al [9] used an asymmetric network design with the Inception module. ResNet proposed by He et al [10] used a residual module to overcome the vanishing gradient problem in earlier CNNs and enabled the expansion of depth of CNNs. The Xception model by Chollet [11] introduced the concept of depthwise convolutions to achieve topology sparsity. The aforementioned

advancements created a succession of more accurate and more specialized CNN models that can be applied for various computer vision tasks.

Currently, one of the primary uses of CNNs in computer vision is image classification. In image classification, the input images are classified into one of many pre-specified categories. Image classification using CNNs has been used to identify whether a given radiology image indicates the presence or absence of a certain disease [12]. Image classification CNNs are also used to identify handwritten digits [13]. Facial recognition, which is assigning a name to a face in an image, is a classification task [14]. Object recognition, which is identifying an object as belonging to an object type category, is an image classification task as well [15]. The key aspect in using image classification CNNs for a certain task is to formulate the task as a classification problem. A related class of tasks is activity detection from videos [16, 17] wherein a specific activity, typically by a human, is detected from a given video. Makantasis et al demonstrated the use of activity detection for industrial activities [18]. Activity detection in beach volleyball was demonstrated by Kautz et al [19]. Complex event recognition in human activity detection aims to detect a sequence of events separated in time [20]. Activity detection is a more complex task compared to image classification. Therefore, a generally preferred method of applying CNN based computer vision techniques is to formulate the task at hand into an image classification problem.

3. Model

We experimented with various classes of neural network model architectures for the objectives of this work. We concluded that a custom convolutional neural network that implements image classification and is trained on a substantial number of baby images in various sleeping positions would be needed for accomplishing the objectives. The three classifications

for the image classification task were: (1) the baby in a back sleeping position (normal) (2) the baby changing from one position to another (3) the baby sleeping in a stomach sleeping position (outside stressor). Our objective is to design a custom CNN architecture that performs the above image classification task with low computational load and low memory footprint. This is to ensure that the model can be embedded in edge devices such as crib cameras while detecting the sleeping position and its changes effectively. The architecture should also be easily trainable and be able to incorporate retraining based on caregiver feedback on the accuracy of the classifications. The core CNN architecture that implements image classification needs to be supported by other components for associated functions such as sending notifications to the caregivers' mobile devices. Our proposed architecture (see Figure 2) draws upon recent advancements in image classification, but at the same time simplifies these so as to address the challenges of computational load and memory footprint at the edge.

In section 3.1, we describe the components of our custom CNN model. In section 3.2, we describe the components that surround the core CNN model in order to provide the full functionality of a complete practical system. In section 3.3, we describe the details of our implementation of the proposed architecture and the experimentation conducted to arrive at the most optimal architecture.

3.1 Model architecture

The overall architecture of our custom CNN can be represented as follows:

Input image ⇒ {Convolutional Layers + Max Pooling Layers} x multiple times ⇒ Fully Connected Layer ⇒ Output.

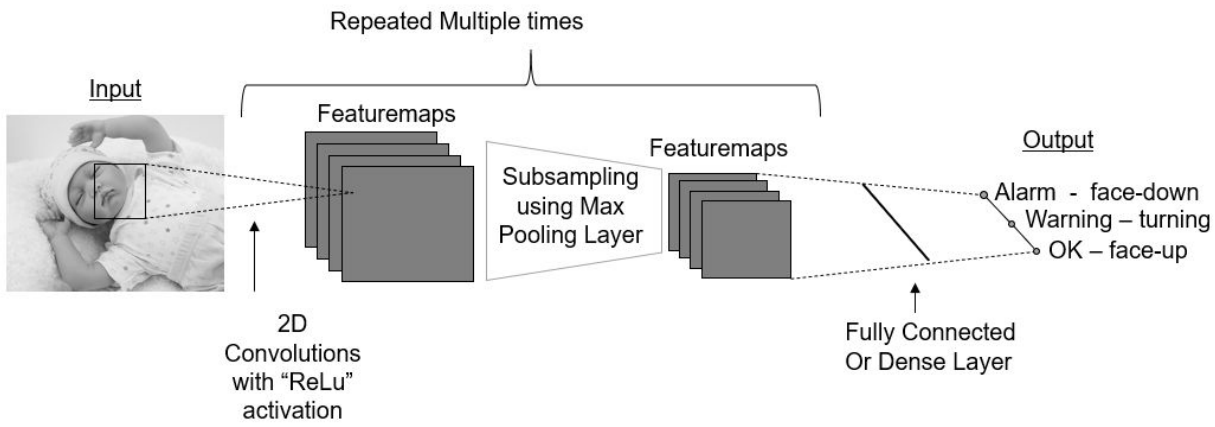

Fig. 2 . Custom CNN system architecture

3.1.1 Input image

A 2D image of the baby captured by the camera is the input data. Color images would have multiple channels. Since our aim is to identify the sleeping position of the baby and this will not have any correlation to the color (as compared to automatically recognizing a traffic signal) grayscale images are sufficient. In addition, due to the computational inefficiency of large color images, which require matrices/data of multiple dimensions to be passed between various layers in the network, grayscale images were used to reduce the computational cost and increase efficiency. Once the input image is converted into the grayscale, it is rescaled to a 100 x 100 size. This was found to be computationally efficient without losing any key features required to recognize the sleeping position of the baby. This preprocessing to grayscale and resizing is performed both for training data as well as live images captured for classification.

3.1.2 Convolutional layer

As the name suggests, convolutional layers are the key building blocks of any CNN. A 2D convolutional layer takes in 2D data as input and produces a 2D data as output. Every data point within the output corresponds to certain features present within a window of the input. For this system, a 3x3 convolutional window was used. The convolutional window’s features are

simplified into a single value by calculating the dot product of the convolutional window with the values of a similar sized weight matrix called a kernel or filter, formed during the model training process (starting with random values and corrected during training by backpropagation). The output of the above convolution is passed through an activation function. We used the Rectified Linear Unit (ReLU) as the activation function due to its ability to eliminate the issue of vanishing gradients in large scale convolutional neural networks. Once the output is calculated, the input window is shifted by the stride value. For this system, a stride value of 1 was used. The calculation continues through various dimensions of the input to produce the output map or the featuremap. Multiple of the filters could be used within a convolutional layer. This system used 32 filters.

3.1.3 Max Pooling layer

While the convolutional layer produces a similar sized feature-map, the Max Pooling layer helps in picking up the most relevant features and reducing the size of the data. Here also a similar moving window concept is applied, and on each window a pooling function is applied. For this system, the window size of 2x2 and a kernel of the same size, with a stride of 2 was used with Max (maximum of the given values) as the pooling function – Max Pooling retains the largest value within each window. This halves the data across the dimensions.

3.1.4 Layers before they converge

The data from previous layers is flattened before being passed to the dense layer, and a dropout rate of 0.5 was used to avoid the overfitting of data passed through the network and therefore increase training accuracy. Dropout is a technique where randomly selected neurons in a given layer of the neural network are shut down (not operational) during training. They are "dropped-out" randomly [20]. Overfitting occurs when the model performs extremely well in

classifying training data but fails to correctly classify and process valid images outside a particular training dataset.

3.1.5 Dense layer to converge the data into the required classes

Dense layers or fully connected layers are the typical multilayer perceptron layers, where all inputs are connected to the required outputs. In the final layer in our implementation, the outputs are the three classes corresponding to the three possible baby positions. In this particular application of dense layers, the Sigmoid activation function was used.

3.1.6 Choosing the right architecture

The combination of a Convolutional Layer with a Max Pooling Layer was used as a single convolutional processing unit. Different models were built and trained with variations in the combinations of

- Number of (Convolutional layer + Max Pooling layer) convolutional processing units
- Number of filters within each Convolutional layer
- Number of Dense layers

The combination that yielded the best result, in terms of a higher accuracy rate combined with the least error/loss, was identified for production deployment. Accuracy is calculated by counting the correct predictions and dividing it by the total number of predictions. The Sparse Categorical Cross Entropy function was used to calculate the loss.

3. 2 Supporting components for end-to-end system

Figure 3 shows the high-level system functionality with end-to-end system components including the above CNN architecture at its core.

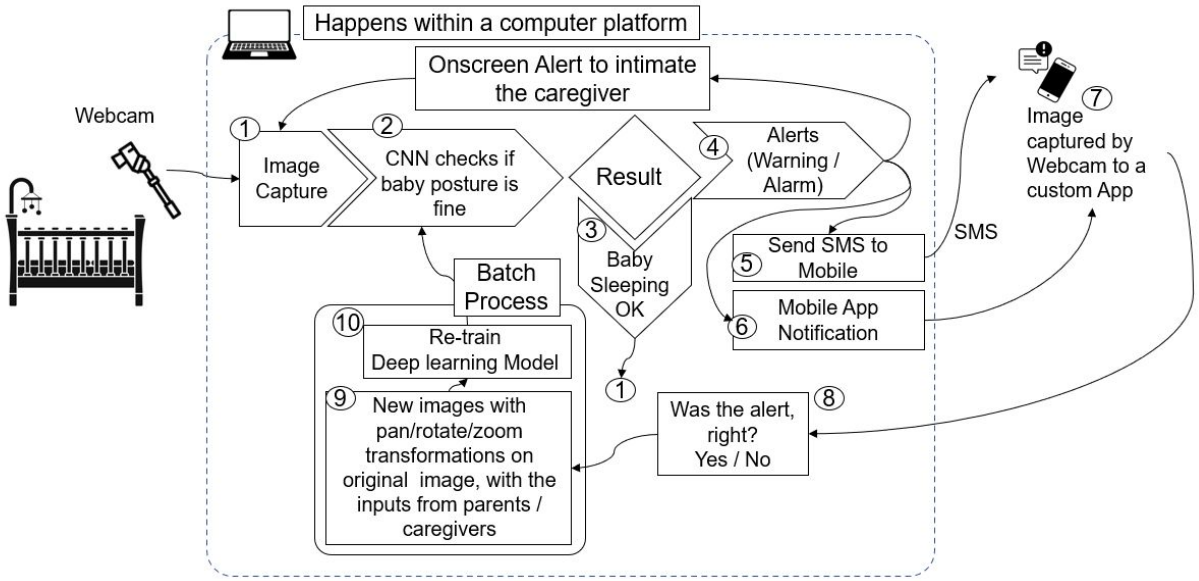

Fig. 3. End-to-end system design

(1) A camera is positioned facing the baby. The camera is connected to a computer (which could be a processing unit that is part of the camera). The program running on the computer is capable of taking pictures.

(2) The high resolution color image from the camera is converted to grayscale, re-sized to 100x100, and passed on to the Convolutional Neural Network architecture described in section 3.1. The model returns confidence levels against the three categories of sleeping positions (Sleeping on Back – OK , Baby trying to turn – Alert, Baby turned over and lying on stomach – Alarm).

(3) For a specific image captured, if the system decides the baby is OK; it moves to Step 1 to capture the next image and proceeds to further steps

(4) In the case when the system decides to notify the caregivers, an onscreen visible indicator and an audible alert goes on.

(5) Also the system can generate multiple channels of alerts such as an SMS text message. These alerts are sent to pre-configured mobile numbers.

(6) Alternatively, a Push Notification can be sent to a mobile application along with the picture from the camera.

The system can generate multiple alerts in multiple channels (for example, SMS to few phones, Push Notifications to multiple mobile devices, voice messages on home automation devices, etc). The intent is to ensure that a channel failure or latency do not hinder the performance of the system.

(7) The Push Notifications can be sent along with the captured baby image, with options for the caregiver to acknowledge and validate the notification. Two options are provided for the caregiver; a) "OK" to acknowledge the notification as correct and switch off alarm/vibration b) "False Alarm" to indicate that the system prediction is incorrect.

(8) Based on the caregiver's acknowledgement that the notification is correct, the corresponding images are annotated to the predicted class. If the caregiver disagrees with the prediction, the prediction with the next higher probability value is taken as the annotation for the next round of training.

(9, 10) Periodic retraining with live images can also be performed. As explained in Step 1, the system continuously captures the baby's images for predicting the baby position. These images are retained for retraining along with the predictions corresponding to these images. Any image where there is an acknowledgement from the caregiver (Step 8) is also included for the retraining. Periodic retraining of the model with the images captured as part of the system execution adapts the system to a specific baby.

3.3 Implementation and experimentation

The custom CNN system was built leveraging Python 3 with Jupyter Notebook as the development environment. The core of the model was built with the use of TensorFlow/Keras. Sequential modeling (where layers are organized sequentially) was used to develop the various layers in the neural network. The following explains the high-level steps in building the system. The three primary steps in the implementation process were:

(1) Preprocessing (which involves image manipulation) and pickling the training dataset images

(2) Model training

(3) Model deployment, experimentation and continual improvement

3.3.1 Preprocessing and pickling the images

The continuous stream of image data was provided by a webcam that was trained on the head and shoulders area of the baby. OpenCV was used to periodically capture images from the video feed to create the training and testing datasets. Lifelike dolls of three different ethnicities were used while training the model. While capturing the images, the angle of the baby and the position of the baby relative to the crib were changed continuously.

A total of 9,000 images were used as seeds, and every image from the video frame was used to generate additional images. The ImageDataGenerator part of Keras image processing package was leveraged for rotating, cropping, and zooming the seed images. Additional transformations involved altering the height and width of each seed image and flipping the seed images horizontally. The exact parameters for the transformations are listed below.

- Image rotation range up to 20 degrees
- Image shift (pan) horizontally/vertically by 10% of the image width/height

- Allowing horizontal flips
- Zoom range of 20%
- Up to 5% shearing of the image

This transformation process was carried out to ensure the success of the neural network model in classifying different sleeping positions regardless of any new positioning of the camera (that would capture the head and shoulders section of the baby), as well as variations in the image quality and camera quality.

Along with the captured seed images, the total data set included about 18,000 images across the three sleeping position categories. 5% of that (about 900) was used as the test set, and the remaining 95% was used as the training set. Following common practice, this was done due to the relatively large size of the complete data set. The images were then randomized and labeled to the three categories (OK-Face Up, Alert-Turning, Alarm-Facedown). All images were converted to grayscale, and scaled down to 100 pixels. The image size was chosen after trial and error between the visible features of the baby's position and optimal size for the model generation. The sized images were pickled (utility to marshal and unmarshal a data structure) for later use. The data was normalized using (x - x.min()) / (x.max() - x.min()). In our case, since the grayscale image had just one channel, we could normalize the grayscale image data by dividing it by 255, changing every pixel value to be between 0.0 and 1.0.

3.3.2 Model training and experimentation

A straw-man model was created to check if the high-level hypothesis of using CNNs to detect the baby sleeping position and its changes held up. The overall architecture of the straw-man model had 3 sequential sets of convolutional processing units, with each set containing the following structure:

- 2D Convolutional layer with 32 kernels of 3x3, Rectified Linear Unit (ReLU) as activation function followed by a Max Pooling layer with kernel size of 2x2 and stride of 2

Following this was a flattening layer that converted the two dimensional output of the last Max Pooling layer into a one dimensional vector that was fed into 2 dense layers. The first dense layer converged the data to 32 classes with ReLU as the activation function. The final dense layer converged the 32 classes to the required 3 classes, each corresponding to a sleeping position category. The final dense layer had a Sigmoid activation function and a dropout rate of 0.5 (50% probability of shutting down randomly chosen nodes in each layer of the neural network).

The model was compiled with the following parameters:

- The loss function used was "Sparse Categorical Cross Entropy", which computes the cross entropy loss between the labels and predictions.
- The Adam optimizer was used (learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-07 and amsgrad = False)
- Capturing the accuracy matrix

The above model was trained with 8 epochs. The straw-man model yielded an average accuracy rate of 0.8971 and an average loss of 0.2876. This gave us the confidence about our starting hypothesis that CNNs can be used to detect baby sleeping positions quite accurately.

Further experimentation with the aforementioned model was conducted to include multiple combinations of number of convolutional processing unit sets, number of convolutional kernels and number of dense layers prior to the final converging dense layer. TensorBoard logging was enabled and a callback was added in the training step to visualize the accuracy levels and loss through various epochs.

3.3.3 Model deployment and continuous improvement

The model with three convolutional processing sets with each containing one 2D Convolutional layer and one Max Pooling layer with 32 kernels in each set and one Dense layer at the end following the three convolutional processing sets yielded the best results in 12 epochs. This combination was selected for production. A batch size of 32 was used to maintain high computational efficiency and lower the overall space/volume requirement for training each model. The system was further optimized based on the live feedback from the caregiver persona during trial runs. The images captured during the live runs were included (along with standard image transformations discussed earlier) in higher proportions for retraining the model.

3.4 Evaluation metrics

The model training and validation was performed with various combinations of number of convolutional processing unit sets, number of convolutional kernels and number of dense layers as discussed earlier. Accuracy along with loss was used to evaluate the performance of the model. This was logged and visualized in TensorBoard. The following graphs show the accuracy and loss functions for various combinations as above. The intent was to find the best performing model in terms of highest accuracy and lowest loss for this classification task. If two models yielded similar accuracy/loss rates through various epochs, the size (in terms of memory used) of the yielded model was considered.

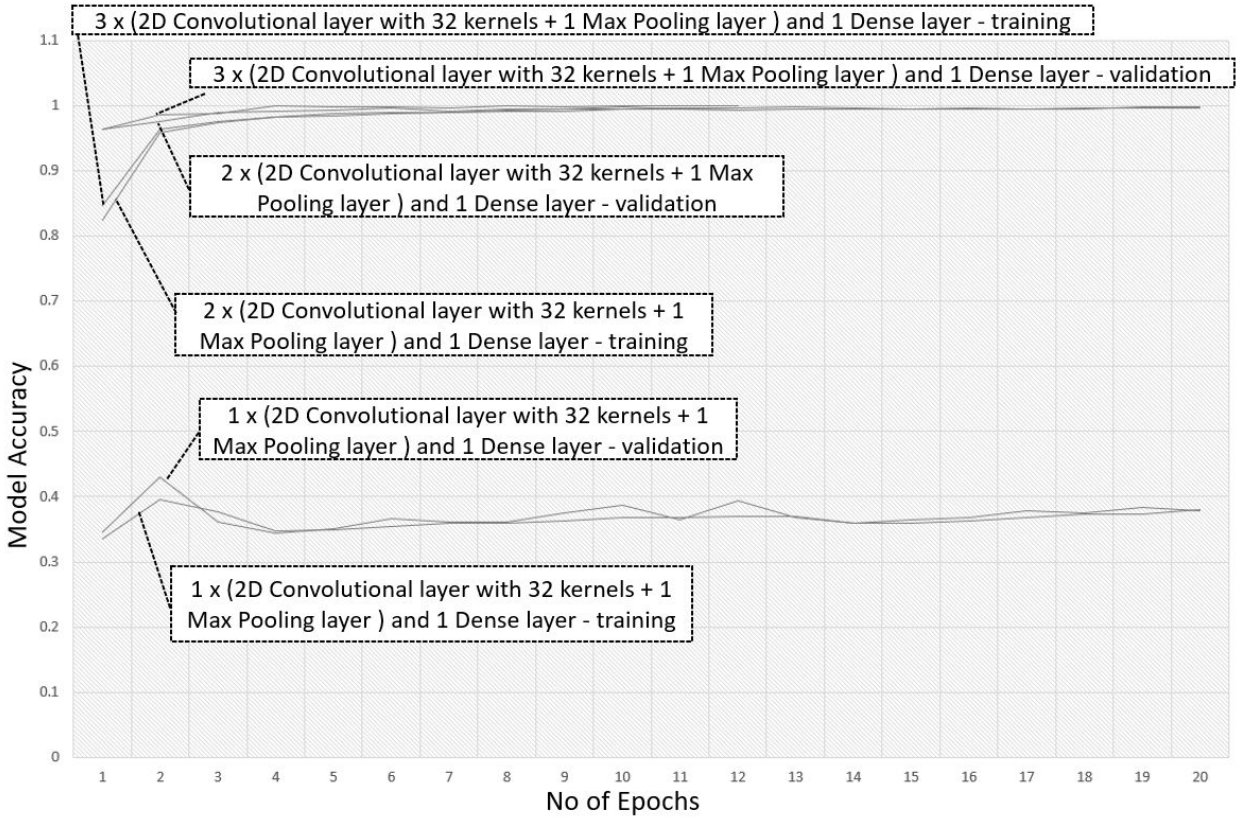

Fig. 4. Accuracy evaluation

Accuracy evaluation: The model with 1 x (2D Convolutional layer with 32 kernels + 1 Max Pooling layer) and 1 Dense layer never improved over an accuracy of 0.4. The rest of the models plateaued around 10 epochs and yielded higher accuracy levels. Various other models such as, 2 x (2D Convolutional layer with 32 kernels + 1 Max Pooling layer) followed by 1 Dense layer and 3 x (2D Convolutional layer with 32 kernels + 1 Max Pooling layer) followed by 1 Dense layer, were executed over 20 epochs, but the models tended to plateau around 10 epochs. Therefore, the number of training epochs was reduced to 12. The models were executed on an Intel i5-8250U CPU @ 1.6GHz, 4 Core, without GPU and 8 GB RAM.

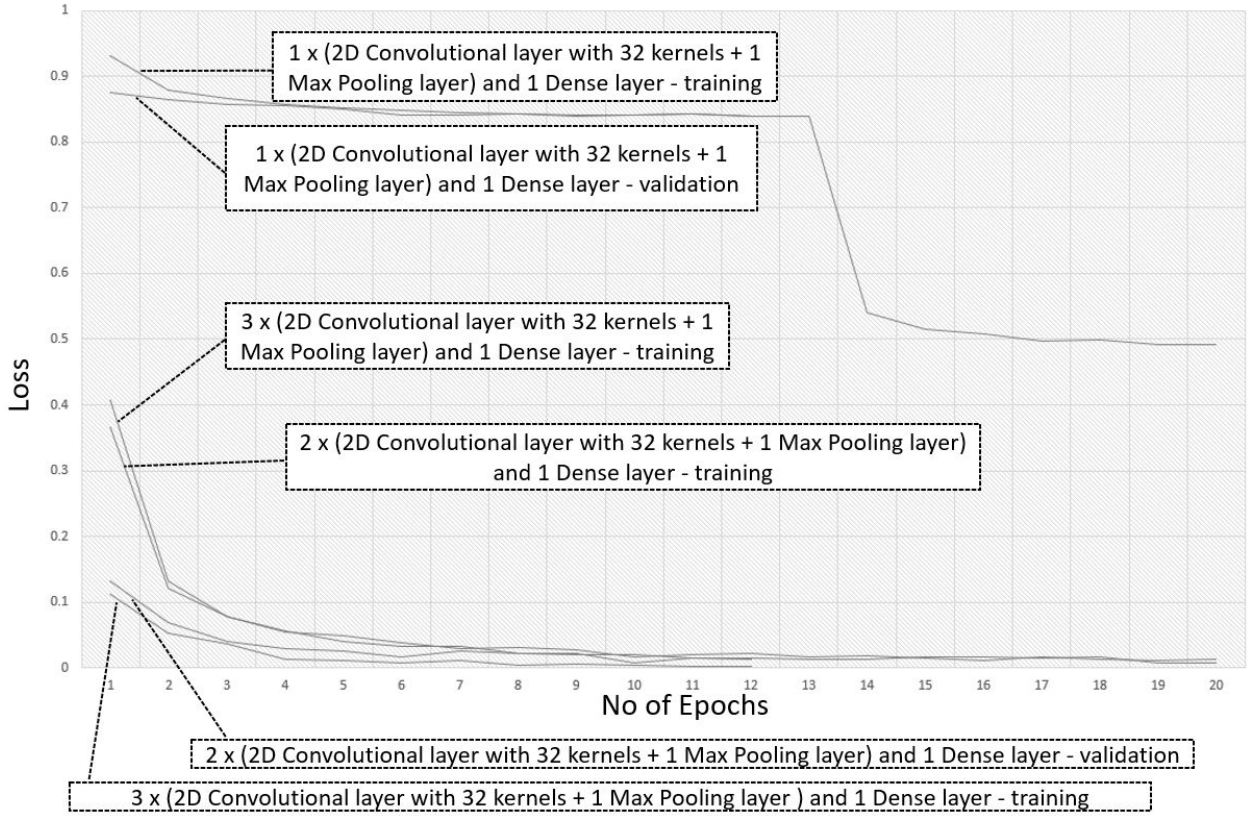

Fig. 5. Loss evaluation

Loss evaluation: The model with 1 x (2D Convolutional layer with 32 kernels + 1 Max Pooling layer) followed by 1 Dense layer had the lowest loss function of 0.5 after 14 epochs. For the other combinations, the loss function considerably reduced after 10 epochs.

The following table summarizes the results of the evaluation of various configurations:

Number of sets of (2D Convolutional layer + Max Pooling layer)	**Number of kernels within the 2D Convolutional layer**	**Number of Dense layers**	**Number of epochs**	**Average time taken for training/ validations**	**Comments/Conclusions**
1	32	1	20	20 Minutes	The training/ validation accuracy never went beyond

					0.4 and loss never reduced from 0.5. This architecture configuration was discarded
2	32	1	20	2 hour 42 minutes	This model yielded an accuracy rate of 0.99 and a loss of 0.02. When the TensorFlow model was saved onto the disk, the size was around 850 KB.
3	32	1	12	25 minutes	This model yielded the highest accuracy rates and low loss after 10 epochs. When the TensorFlow model was saved on disk, the size was 500 KB. With the lowest size, high accuracy, and low loss, this model was selected for production deployment
3	32	2	12	25 minutes	Experimented with an additional dense layer. This model had high accuracy rates and low loss after 10 epochs. The TensorFlow model when saved on the disk had a size of 1.5 MB
3	64	2	12	52 minutes	Experimented with an additional dense layer and double the number of kernels. This model had high accuracy rates and low loss after 10 epochs. The TensorFlow model when saved on disk had a size of 5.7 MB

Summary of Evaluations: The model with 3 repetitions of (2D Convolutional layer with 32 kernels + Max Pooling layer) followed by 1 converging Dense layer was selected for deployment.

3.5 Qualitative results

The production version of the system used three continuous frames captured at a time and these were used to predict the sleeping position of the baby. The results were evaluated as a majority voting combination of the three predictions. During the live test runs, the system always predicted the right posture, especially for the face-up and face-down positions (i.e., the accuracy rates were close to 100%). The processing time taken for the system to capture the three continuous images, transmit the data to the model, classify all three images, and generate corresponding notifications locally and on the mobile (in case of the baby turning or is facedown) was within 3 seconds.

3.6 Runtime evaluation

The finalized production model 3 x (2D Convolutional layer with 32 kernels + Max Pooling layer) followed by 1 Dense Layer had a low memory footprint (the amount of disk space taken was around 500 KB) and took a short time to get trained/retrained (less than 25 minutes). This model could be deployed on low powered single board computers, for example:

- Raspberry Pi: The TensorFlow model could be converted to TensorFlow light and run on Raspberry Pi 4B. This would still perform according to the required standards.
- NVidia Jetson Nano: When using JetPack as the OS on NVIDIA Jetson Nano, TensorRT is pre installed as containers. The model could be converted to TensorRT and deployed on Jetson Nano.

4. Conclusion

We have proposed a custom convolutional neural network computer vision system to help prevent the Sudden Infant Death Syndrome. It can provide alerts via mobile notifications to help avoid the sleeping position outside stressor. The system has been developed to minimize computational load and memory footprint, which enables it to be embedded in edge devices, such as crib cameras. Our experiments show that a model with three sets of 2D Convolutional layer with 32 kernels and a Max Pooling layer, followed by one fully connected dense layer is able to detect baby position changes with very high accuracy in experimental settings and alert caregivers on mobile devices within three seconds. The memory footprint of the model was around 500 kilobytes. This gives us confidence that this custom convolutional neural network model is a viable candidate for consideration as a non-intrusive, low cost technology to help prevent the Sudden Infant Death Syndrome. The system presented in this paper can be further evaluated with live video feeds, enhanced and optimized for deployment in real-world settings.

REFERENCES

[1] National Center of Child Health and Human Development, "Research on Possible Causes of SIDS", Retrieved on December 12 2020 from https://safetosleep.nichd.nih.gov/research/science/causes#f1

[2] National Center of Child Health and Human Development, "What is SIDS?", Retrieved on December 12, 2020, from http://www.nichd.nih.gov/health/topics/sids/conditioninfo/pages/causes.aspx#f1.

[3] A. Wehrli, "SIDS: The 10 Best Prevention Products and 5 That Were Recalled", Baby Gaga, May 26 2018, Retrieved on December 12 2020 from https://www.babygaga.com/sids-the-10-best-prevention-products-and-5-that-were-recalled/

[4] U.S Food and Drug Administration, "Baby Products with SIDS Prevention Claims", October 17 2019, Retrieved on December 12 2020 from https://www.fda.gov/medical-devices/products-and-medical-procedures/baby-products-sids-prevention-claims

[5] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification With Deep Convolutional Neural Networks", Advances in Neural Information Processing 25, 2012.

[8] M. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks", European Conference on Computer Vision, ECCV 2014, pp. 818-833.

[9] C. Szegedy, W. Liu, Y. Jia, et al, “Going Deeper with Convolution”, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[10] K. He, X. Zhang, S. Ren et al, “Deep Residual Learning for Image Recognition”, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[11] F. Chollet, “Xception: Deeplearning with Depthwise Separable Convolution”, arXiv, Apr. 2017.

[12] S. Soffer, A. Ben-Cohen, O. Shimon, et al., “Convolutional Neural Networks for Radiologic Images”, Radiology, 2019, vol. 209, pp. 590-606.

[13] S. Ahlawat, A. Choudhary, A. Nayyar, et al., “Improved Handwriting Recognition Using Convolutional Neural Networks”, Sensors 2020, June 2020.

[14] M. Coskun, A. Ucar, O. Yildrium, et al., “Face recognition based on convolutional neural network”, 2017 International Conference on Modern Electrical and Energy Systems (MEES), Nov. 2017.

[15] Z. Zhao and S, Xu, “Object Detection with Deep Learning: A Review”, IEEE Transactions on Neural Networks and Learning Systems, vol. 99, pp. 1-21, Jan. 2019.

[16] A.S.Voulodimos, D.I.Kosmopoulos, N.D.Doulamis, and T.A. Varvarigou, “A top-down event-driven approach for concurrent activity recognition,” *Multimedia Tools and Applications*, vol. 69, no. 2, pp. 293–311, 2014.

[17] A.S.Voulodimos, N.D.Doulamis, D.I.Kosmopoulos, and T.A. Varvarigou, “Improving multi-camera activity recognition by employing neural network based readjustment,” *Applied Arti- ficial Intelligence*, vol. 26, no. 1-2, pp. 97–118, 2012.

[18] T. Kautz, B. H. Groh, J. Hannink, U. Jensen, H. Strubberg, and B. M. Eskofier, “Activity recognition in beach volleyball using a DEEp Convolutional Neural NETwork: leveraging the

potential of DEEp Learning in sports," *Data Mining and Knowledge Discovery*, vol. 31, no. 6, pp. 1678–1705, 2017.

[19] K. Tang, B. Yao, L. Fei-Fei, and D. Koller, "Combining the right features for complex event recognition," in *Proceedings of the 2013 14th IEEE International Conference on Computer Vision, ICCV 2013*, pp. 2696–2703, Australia, December 2013.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting". *J. Mach. Learn. Res*., vol. 15, no. 1, 1929–1958, January 2014.