Journal of Defense Analytics and Logistics

143

Strategies to alleviate flickering: Bayesian and smoothing methods for deep learning classification in video

Noah Miller, Glen Ryan Drumm, Lance Champagne and Bruce Cox
Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, and
Trevor Bihl
Air Force Research Laboratory, Wright-Patterson AFB, Ohio, USA

Received 12 September 2023
Revised 21 May 2024
23 August 2024
21 September 2024
Accepted 23 September 2024

Abstract

Purpose – Increasing reliance on autonomous systems requires confidence in the accuracies produced from computer vision classification algorithms. Computer vision (CV) for video classification provides phenomenal abilities, but it often suffers from "flickering" of results. Flickering occurs when the CV algorithm switches between declared classes over successive frames. Such behavior causes a loss of trust and confidence in their operations.

Design/methodology/approach – This "flickering" behavior often results from CV algorithms treating successive observations as independent, which ignores the dependence inherent in most videos. Bayesian neural networks are a potential remedy to this issue using Bayesian priors. This research compares a traditional video classification neural network to its Bayesian equivalent based on performance and capabilities. Additionally, this work introduces the concept of smoothing to reduce the opportunities for "flickering."

Findings – The augmentation of Bayesian layers to CNNs matched with an exponentially decaying weighted average for classifications demonstrates promising benefits in reducing flickering. In the best case the proposed Bayesian CNN model reduces flickering by 67% while maintaining both overall accuracy and class level accuracy.

Research limitations/implications – The training of the Bayesian CNN is more computationally demanding and the requirement to classify frames multiple times reduces resulting framerate. However, for some high surety mission applications this is a tradeoff the decision analyst may be willing to make.

Originality/value – Our research expands on previous efforts by first using a variable number of frames to produce the moving average as well as by using an exponentially decaying moving average in conjunction with Bayesian augmentation.

Keywords Computer vision, Bayesian neural networks, Video classification

Paper type Research paper

1. Introduction

The technological nature of warfare is rapidly evolving with ever-increasing emphasis placed on processes driven by large swaths of data. Naturally, as the complexity of the command and control (*C2*) decision space grows, the speed of command structures becomes a limiting factor. As a result, the US Department of Defense (DoD) is increasingly relying on autonomous systems and artificial intelligence techniques that drive autonomous decision-making. The 2022 National Defense Strategy (NDS) explicitly directs investment in "militarily-relevant

The authors would like to thank the Air Force Research Laboratory for funding in support of this research. Funds were utilized to support student computer services, student and faculty attendance to professional conferences, as well as partially fund faculty developmental salary.

Disclaimers: All opinions are solely those of the authors and do not reflect the Air Force Institute of Technology, the United States Air Force or the Department of Defense.

Journal of Defense Analytics and Logistics
Vol. 8 No. 2, 2024
pp. 143-159
Emerald Publishing Limited
2399-6439
DOI 10.1108/JDAL-09-2023-0010

capabilities in trusted artificial intelligence and autonomy" (Office of the Secretary of Defense (OSD), 2022). Data-driven decisions are reliant on intelligence, surveillance and reconnaissance (ISR) assets, and the predominant movement in ISR assets is toward sensor fusion and automation (Bihl and Talbert, 2020; Andrejevic, 2019). Key among the ISR data produced is video and the accompanying need for activity classification (McCloskey *et al.*, 2023).

Video classification typically takes one of two forms, activity classification (Karpathy *et al.*, 2014) or object detection and classification (Bochkovskiy *et al.*, 2020). However, even most of the advanced approaches within the state-of-the-art and state-of-the-practice consider each frame as an independent observation (Xu *et al.*, 2021; Shah *et al.*, 2021; Kurian *et al.*, 2020). Naturally, this is an incorrect assumption and by independently categorizing each frame, the linkage between past and current frames is severed. This broken link prevents the model from using all of the information available.

The result of this incorrect assumption of independence is that video-based image classification models have a critical, unwanted effect often referred to as "flickering" in the literature (Xu *et al.*, 2021; Shah *et al.*, 2021; Kurian *et al.*, 2020). Flickering occurs when the algorithm switches back and forth between declared classes over successive frames. This can occur as a correct prediction on an object switching to a false prediction for a frame or two, and then switching back to the correct classification. Flickering can be due to a variety of causes, such as occlusion, changes in lighting or camera angle, random variations in the image or even random failures of algorithm.

In situations and scenarios where flickering occurs, users invariably ask one of the typical questions of automation (Bihl *et al.*, 2020):

(1) What is it doing?

(2) Why is it doing that?

(3) What will it do next?

In general, these questions come about due an automation (or AI/ML) solution having a problem in handling one of the "ilities:" trustability, utility, reliability, explainability, etc. (Bihl *et al.*, 2020). And such "ilities" are of increasing importance as AI algorithms operate safety- and privacy-critical processes: autonomous vehicles, facial recognition, etc.

To address issues with such "ilities" Swize *et al.* (2022) explores Bayesian neural networks, providing detailed model prediction probabilities and increasing overall model accuracy. However, Swize *et al.* (2022) uses entire video segments and is not useful for real-time processing of frame-based data. Extending this foundation, this paper considers the following research questions:

RQ1. Can a Bayesian approach to handling probabilities from video-based classification improve flickering?

RQ2. Can flickering of video-based classification results be smoothed to provide consistency?

This paper is organized as follows: Section 2 presents background on artificial neural networks (ANN) and flickering; Section 3 presents the video dataset and our novel algorithmic approaches; Section 4 includes results and examples of the approach; Section 5 provides conclusions.

2. Neural networks and computer vision

Computer vision (CV) is a rapidly growing field of computer science that applies AI/ML to provide for pattern recognition and item/task classification in images or video. Due to significant progress in ANNs, such as deep learning, and graphics processing unit (GPU)

computing power the performance of computer vision techniques has rapidly excelled since the 2000s (Ball *et al*., 2017).

ANNs, including deep learning, are machine learning models inspired by neuroscience principles and information networking (Bihl *et al*., 2020). While the basic unit of ANNs are simplistic threshold logic units (TLU) (Géron, 2023), ANNs see considerable advantages when networks of multiple TLUs are made whereby their interconnection enables increasing powers of inference. Multilayer perceptrons (MLP), as conceptualized in Figure 1, are the result of such combinations and in an MLP, the middle layers are referred to as the hidden layers and the last layer is referred to as the output layer.

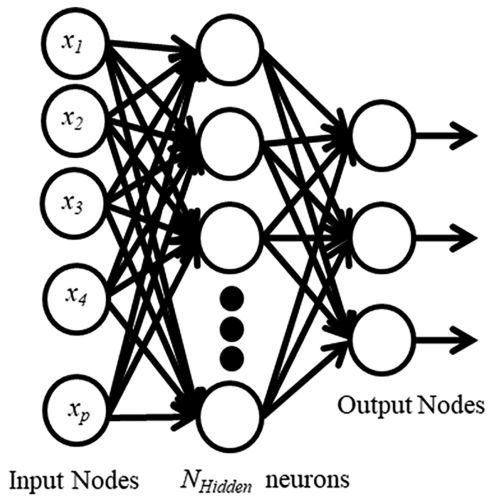
Computational ANNs, as conceptualized in Figure 1, are biologically inspired and consist of multiple interconnected nodes termed "neurons" (Bihl *et al*., 2020). However, rather than the complex communication that occurs in biology, through molecular, electrical, cellular, systems and behavioral means (Sweatt, 2016; Kandel *et al*., 2021), artificial neurons are considerably simpler, employing statistical methods to learn patterns between inputs and outputs (Jain *et al*., 2000). Through organizational and iterative principles, connection weights between neurons, inputs, outputs and interconnected hidden layers are computed to learn nonlinear relationships in data (Jain *et al*., 2000).

When expanded with multiple hidden layers, typically four or more, ANNs begin to become "deep" and are often termed deep neural networks (DNNs). Using their large amount of interconnected nonlinear functions, DNNs possess intrinsic abilities for automated feature extraction although at relatively large computational cost. As computing expanded in the 2000s DNNs led to revolutionary gains in computer vision (CV), speech recognition and other applications. While there are multiple DNN architectures, four primary types exist (Ball *et al*., 2017):

(1) Autoencoders (AE)s, ANNs for unsupervised data exploration

(2) Deep belief networks (DBNs), probabilistic graphical models

(3) Convolutional neural networks (CNNs), which mimic biological visual data exploitation through convolutions, pooling and nonlinear functions



Source(s): Bihl *et al*. (2020)

Figure 1. Basic conceptualization of an ANN with input, hidden and output layers

(4) Recurrent neural networks (RNNs), temporal approaches.

Given their applicability, the DNNs of interest herein for CV applications are CNNs.

2.1 Convolutional neural networks

CNNs are a variant of DNNs that extend from the ideas of LeNet-5 (LeCun *et al.*, 1998), which introduced convolutional layers and pooling layers, the backbones of the modern-day CNN. In a general CNN, convolutional layers are trained to filter and process "neighborhoods" of an image which implies that each neuron's area of influence is based on its location. The combined result of multiple filtering layers in a CNN lends itself well to image and video classification (Géron, 2023). With successive convolutional and pooling layers, CNNs assemble simple features into increasingly more complex features with each hidden layer (Géron, 2023). At the end of the CNN, a final classification layer usually exists which is often a single hidden layer of a traditional ANN (Bihl *et al.*, 2020).

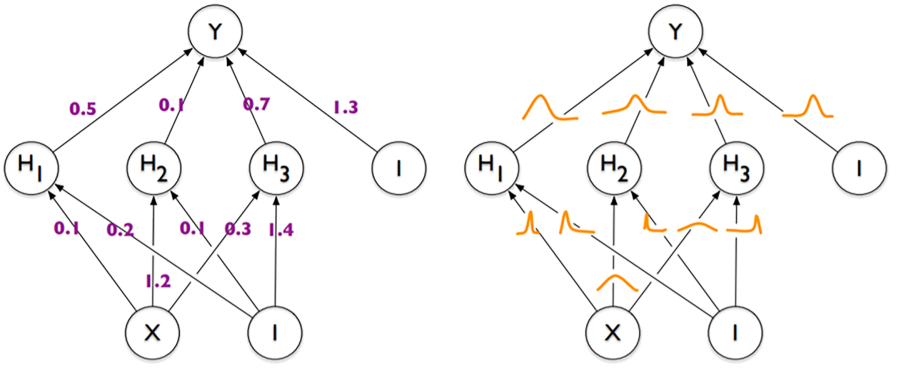
CNNs which include a temporal channel, such as those introduced by Karpathy *et al.* (2014), provide several intriguing approaches to connect frames and learn spatial and temporal features of videos. However, while the approaches fuse information from multiple frames they are focused on classifying video clips versus on the real-time classification of an incoming video feed.

2.2 Bayesian neural networks

A pivotal shortcoming of traditional neural networks is that they have no way to express their level of uncertainty, resulting in seemingly overconfident predictions. Bayesian neural networks (BNNs) quantify uncertainty using the weights in the neural network, replacing weight point estimates with distributions, from which random samples are drawn. An example of weight estimates for a neural network and weight distributions for a BNN can be seen in Figure 2.

Tishby *et al.* (1989) introduces the BNN, addressing the issue that neural networks often perform well on training data but generalize poorly to test data which are not from the exact same distribution as the training data. This approach measures performance outside of the training set, using only the training set, by utilizing the average statistical prediction error. This information is used to develop an optimal network architecture.

BNNs often consider variational inference (VI) when approaching the concept of backpropagation. VI transforms backpropagation into an optimization problem for BNNs utilizing, most frequently, the Kullback-Liebler (KL) divergence (Kullback and Leibler,



Source(s): Blundell *et al.* (2015)

Figure 2. Point estimates compared to distributions on weights

1951). KL divergence measures the difference between the variational distribution and the true posterior. Equation (1) shows the (KL)-divergence:

$$KL(q_\theta(\omega)|p(\omega|D)) = q_\theta(\omega)\log \int \frac{q_\theta(\omega)}{p(\omega|D)} d\omega \quad (1)$$

where $q_\theta(\omega)$ is the density over the set of latent parameters ω. This distribution is limited to a family of simple distributions which can be parameterized by θ (often Gaussian). The true posterior is given by $p(\omega|D)$, where D represents the observed data.

The VI approach was not widely used in the literature in almost two decades, which was, according to Graves, "due to the difficulty of deriving analytic solutions to the required integrals over the variational posteriors" (Graves, 2011). The key development that made training BNNs tractable was the development of "Bayes by Backprop" (Blundell *et al*., 2015). This approach derives unbiased estimates of the derivative of an expectation. By using reparameterization to find the expected lower bound of a given function Bayes by Backprop minimizes the KL divergence between the approximate and true posterior.

Gal and Ghahramani (2016) implements Bayesian thinking into convolutional neural networks, proving that under specific circumstances utilizing dropout in the forward pass of a CNN, instead of only during training, is mathematically identical to a BNN. Further, this method shows a considerable improvement when compared to traditional neural networks regarding classification accuracy. Goan and Fookes (2020) add Bayesian thinking to neural networks, implementing the Bayes by Backprop algorithm to the kernels of a CNN, specifically LeNet-5. Compared to the regular LeNet-5, the Bayesian version possesses the benefit of providing uncertainty levels on predictions, at the cost of only a marginal drop in accuracy.

Finally, Wen *et al*. (2018) develops the method utilized in this paper, called Flipout, stating that weight perturbation algorithms, such as Bayes by Backprop, suffer from high variance of gradient estimates since all the mini-batch samples have the same perturbation, which causes correlation between gradients. The "Flipout layer" uses a flipout gradient estimator to minimize the KL divergence up to a constant: the "negative evidence lower bound". This technique perturbs weights independently within each mini batch. We leverage Flipout layers within our implementation of Bayesian CNNs.

2.3 Prior work on classification flickering

There are several notable papers which address the topic of classification flickering; these techniques all broadly fall under the umbrella of smoothing. Kurian *et al*. (2020) utilizes a moving average of the outputs from a CNN to help eliminate flickering when making predictions on videos of surgery. The approach utilizes a standard CNN, with only one prediction per frame, throughout the video. These predictions are averaged over the previous 16 frames to give the final prediction. Similarly, Stamoulakatos *et al*. (2021) uses a simple moving average of CNN predictions. This approach shows that a simple moving average of CNN predictions can outperform a 3-dimensional CNN, which requires an entire video as an input. These 3-dimensional CNNs cannot be utilized with live video in real time as they need all video frames, including future frames not available in real time. It would seem intuitive that a method with access to all frames would perform better than a moving average, however this result shows the opposite for their data.

In comparison to these prior approaches to reduce video classification flickering our research expands on that of Kurian *et al*. (2020) and Stamoulakatos *et al*. (2021) by first using a variable number of frames to produce the moving average (vice the fixed 16 frames in Kurian) as well as by using an exponentially decaying moving average in conjunction with Bayesian augmentation.

3. Methodology

To address the research questions posed in Section 1 we explore how standard CNNs experience flicker on a large dataset of videos, UCF101. We then develop two approaches to address this flickering: first, through the augmentation of Bayesian layers to the CNN model and second through a smoothing approach. The following sections detail the video data and both approaches in full detail.

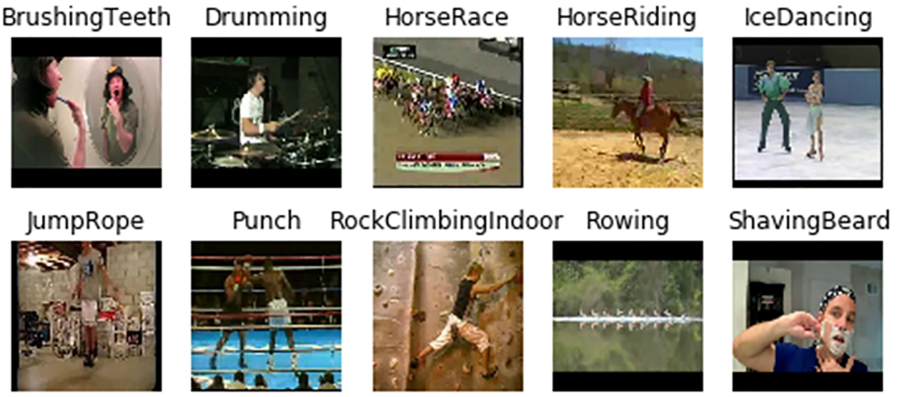
3.1 Representative video data

For a large representative dataset of videos, the UCF101 Action Recognition Data Set was used (Soomro *et al.*, 2012). The UCF101 was introduced in 2012 and was the largest to date compilation of human action videos. UCF101 comprises 13,320 videos across 101 classes of actions. Each class of action is further divided into 25 groups (each containing approximately 4–7 videos) based on common features, such as camera effects, viewpoint, pose and scale, background and lighting conditions. While 101 action classes were captured, these are grouped into five types: human-object interaction, body-motion only, human-human interaction, playing musical instruments and sports.

Due to computational limitations only 10 classes were utilized in each of the classification methodologies: Bayesian and smoothing. The Bayesian approach examined brushing teeth, drumming, horse race, horse riding, ice dancing, jump rope, punch, rock climbing indoor, rowing and shaving beard. Examples of each of these classes can be seen in Figure 3 – smoothing utilized applying lipstick, archery, brushing teeth, cutting in kitchen, fencing, golf swing, head massage, hula hoop and pizza tossing.

Consistent with the preprocessing of (Swize *et al.*, 2022), for processing and preparation for training and testing, each video is partitioned into individual frames sized 224 x 224 pixels, with each pixel containing values for 3 color channels. The images are converted to three-dimensional arrays, sized 224 x 224 x 3. These arrays are normalized by dividing all values by 255 to have all pixel values range from 0 to 1.

All video clips from the 20 classes included in this study were utilized either for training or testing. A total of 80% of the video clips from each class were utilized in training, while the remaining 20% were set aside for testing.



Source(s): Authors' own work

Figure 3. Reference classes from UCF101

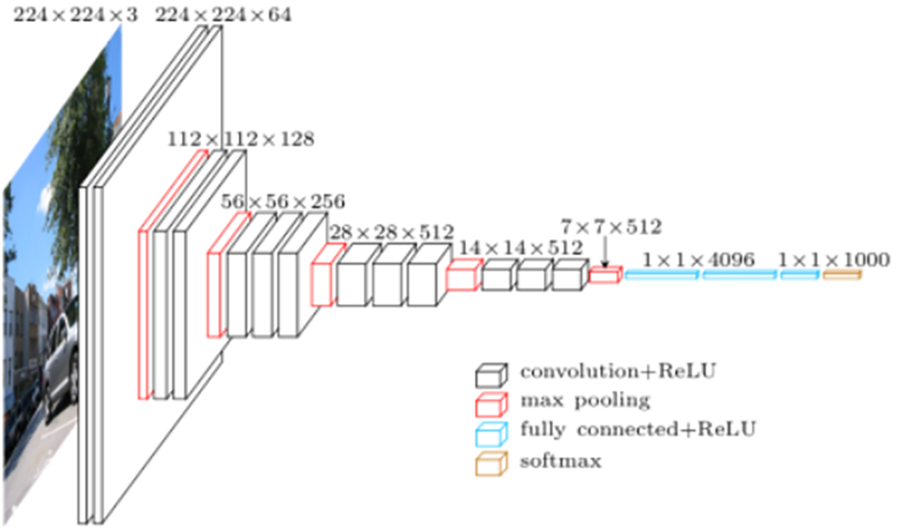
3.2 Bayesian CNNs with decaying weights

To investigate RQ1, i.e., the potential benefits of adding Bayesian concepts to traditional CNNs, as applied to video classification flickering, we developed two models, both based on the VGG16 architecture (Simonyan and Zisserman, 2015), see Figure 4.

The first (i.e., non-Bayesian baseline) model matches the VGG16 architecture with the following pragmatic modifications: instead of the fully connected layers (in blue in Figure 4) having 4,096, 4,096 and 1,000 neurons, respectively, we reduced them to have 256, 128 and 10 neurons. These changes were necessary since the original VGG16 model was developed to classify the 1,000 classes within ImageNet while our experiment contains only 10 classes; hence the original number of neurons would have severely overfit our reduced dataset. We also created a second, Bayesian, version of this architecture by replacing the final dense layer with a 10 neuron dense Flipout layer which implements an approximation of the distribution; acting as a Bayesian dense layer analog with Gaussian priors (Wen *et al*., 2018).

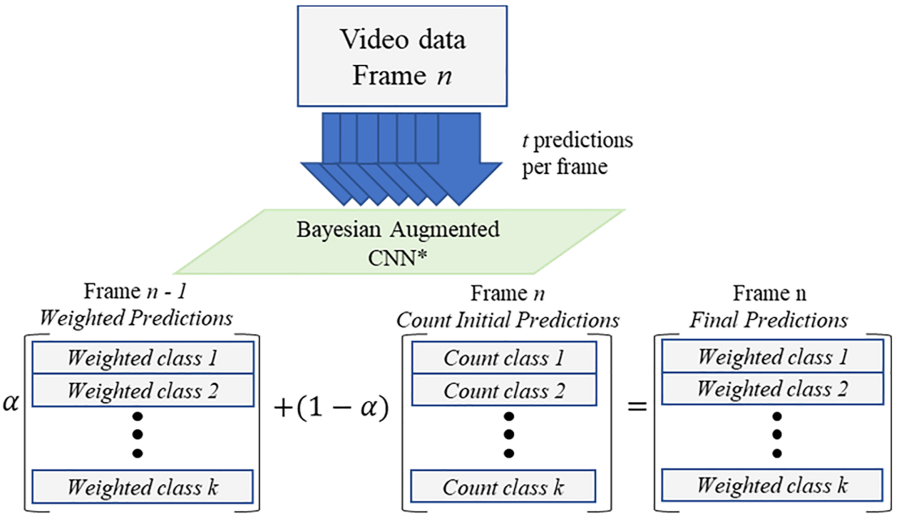
The process for the Bayesian version is presented in Figure 5 and described in detail below. Figure 5 shows, for k classes, the process for transition from raw output for frame n to a weighted classification for frame n. At the end of this process the frame is classified according to the class with the highest weighted count.

Video classification for the traditional CNN is performed independently for each frame. The frame is classified according to the highest probability softmax output from the 10 classes. In comparison a frame's classification for the Bayesian augmented CNN is calculated through a weighted average of both the current frame's 10 softmax outputs along with all prior frames' softmax outputs. Thus, the classification of the current frame is based on an exponentially decaying weighted average which uses two tunable parameters: (1) the number of predictions per frame, t; and (2) a hyperparameter α (the smoothing weight) which controls the exponentially decaying weight applied per frame. As shown in Equation (2) frame n's final prediction vector (Y_n^{final}) is based on both the softmax outputs from frame n (Y_n^{init}, weighted by $1-\alpha$) as well as the final prediction vector from the prior frame's outputs (Y_{n-1}^{final}, weighted by α).



Source(s): Authors' own work

Figure 4. Architecture of the original VGG16 model from Simonyan and Zisserman (2015)



Source(s): Authors' own work

Figure 5. Bayesian augmented CNNs with exponentially decaying weighting process for frame *n*

$$Y_n^{final} = \alpha Y_{n-1}^{final} + (1-\alpha) Y_n^{init} \tag{2}$$

For concreteness consider the following example where we sample BNN weights and predict $t = 20$ times for each frame, smoothing weight $\alpha = 0.25$, and there are 10 classes. If $Y_n^{init} = [0, 10, 0, 0, 0, 0, 0, 0, 0, 10]$, (i.e. frame n predicts class 2 10-times and class 10 10-times) and $Y_{n-1}^{final} = [0, 19, 0, 0, 0, 0, 0, 0, 0, 1]$ then by Equation (2):

$$\begin{aligned} Y_n^{final} &= 0.25\, Y_{n-1}^{final} + 0.75\, Y_n^{init} \\ &= [0, 12.25, 0, 0, 0, 0, 0, 0, 0, 7.75] \end{aligned} \tag{3}$$

where Y_n^{final} is used to classify frame n (in this case as class 2). Similarly, in an iterative fashion, Y_n^{final} is used during the computation of Y_{n+1}^{final}. Thus the current frame's classification uses an exponentially decaying moving average of all prior frame classifications with a user-tunable hyperparameter controlling the smoothing weight applied to the current frame (α) versus the weight applied to the prior frames $(1-\alpha)$.

Table 1 provides several cases of effective weights for the current frame n's vector as well as the effective weight for the five prior frames' vector, for $\alpha = 0.75, \alpha = 0.50, \alpha = 0.25$, and $\alpha = 0.10$. Note that 1) $\alpha = 1$ indicates all weight is concentrated on the current frame's vector (i.e. the network reduces to the standard CNN) and 2) as $\alpha \to 0$, proportionally more weight is placed on prior frames' vectors.

The second tunable parameter, t, is the number of predictions computed each frame, which is directly related to the sample size selected from the distribution of BNN weights. Increasing t leads to increasing estimation precision. This comes at a computational cost, realized as a lower real-time capability of frames per second for the resulting video classification model. We explore hyperparameter tuning for both α and t in Section 4.

Table 1. Relative weights for various values of alpha (α)

Frame	Alpha α = 0.75	α = 0.50	α = 0.25	α = 0.10
n	0.750	0.500	0.250	0.100
n-1	0.188	0.250	0.188	0.090
n-2	0.047	0.125	0.141	0.081
n-3	0.012	0.063	0.105	0.073
n-4	0.003	0.031	0.079	0.066
n-5	0.001	0.016	0.059	0.059

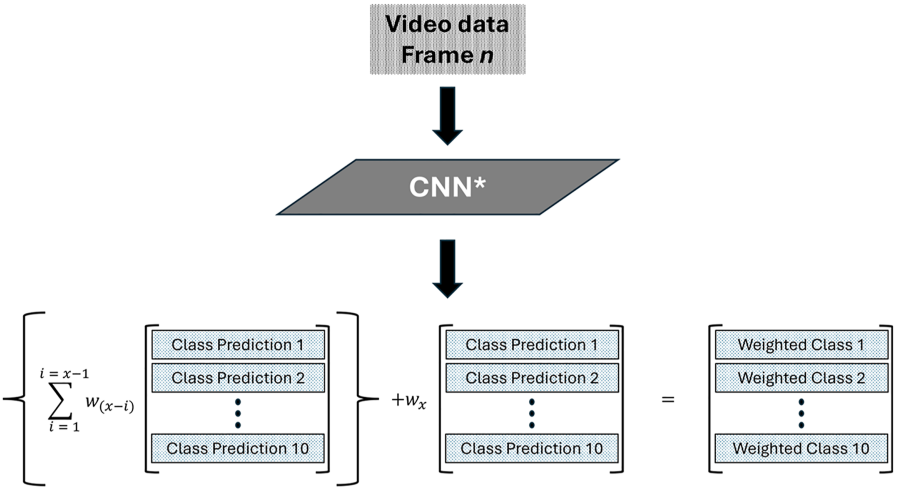
Source(s): Authors' own work

3.3 CNN classification smoothing

To investigate RQ2, the methodology described in this section details the usage of a classification smoothing technique to mitigate video classification flickering. Resnet-50 was used as the base model for this experiment, and transfer learning was used to freeze part of this network (He *et al.*, 2016). The first 49 of Resnet-50's 50 layers were frozen. The last layer of the Resnet-50 network was removed and replaced with an average pooling layer (7x7) that is flattened to a single dimension vector. This is then fed into two dense layers to reduce the dimensionality of the output to 10 classes corresponding to the data. The first dense layer has 512 nodes with ReLU activation and 50% dropout to reduce overfit, and the final (output) layer has 10 nodes with softmax activation.

As previously stated, CNNs perform video classification by treating each frame independently. CNNs are especially useful for showcasing the video flickering at the frame level (Géron, 2023). In this traditional mode, CNNs use no temporal information in classification. Figure 6 shows the use of independent classifications from multiple recent frames to compute a weighted classification for frame *n*.

This technique allows the CNN's independent prediction for the current frame, as well as previous frames, to hold some weight in the current class prediction. To do this, a weighted



Source(s): Authors' own work

Figure 6. Weighted prediction vector process for frame *n*

average smoothing technique is used, where smoothing is performed over some window size, length x.

Every frame's independent prediction vector is computed by the CNN in a typical fashion using a softmax activation. Thus, the independent prediction vector for a frame is a pseudo probability distribution output from the CNN.

The smoothed prediction vector for frame n is calculated using a weighted average of the independent prediction vectors for frame n and the previous $x-1$ frames. As a result, there are two hyperparameters utilized by this method, the window size, x and the weights, $w_1, \ldots, w_x$. For example, for $x = 5$ frames, the final smoothed classification vector is, $Y_5^{final} = [\sum_{i=1}^{5} w_i Y_{i,1}, \sum_{i=1}^{5} w_i Y_{i,2}, \ldots, \sum_{i=1}^{5} w_i Y_{i,10}]$, where $\sum_i^5 w_i = 1$.

Importantly, the smoothed prediction vector is not used for future smoothed predictions (the independent vectors are used). This minimizes the opportunity for getting stuck in a recurring prediction of the same class because of past smoothing.

Window size can be changed to produce results with different characteristics. Shorter windows result in less past information being used but provides for a more responsive network, while the opposite is true for longer windows.

4. Analysis and evaluation

This section provides examples of the benefits of a BNN for video classification, and weighted smoothing across frames; both are considered separately with the UCF101 data.

4.1 CNN classifications with Bayesian augmentation

Using the approach in Section 3.2, both the traditional CNN and the Bayesian augmented CNN were initially trained four times each on 80% of the training data and results validated on the remaining 20%. The goal of this initial training was to test the performance of four different optimizers: Adam, NAdam, Adamax and Adagrad. The accuracy results against the validation set can be seen in Table 2. On both the traditional CNN and the Bayesian augmented CNN the Adamax optimizer performed the best and was chosen for training using the full training set.

Various levels of both the weight decay parameter α and the number of predictions per frame t were explored to determine their impact on both accuracy and flickering. Four levels of α were tested, each with 20 predictions per frame: $\alpha = [1, 0.75, 0.5, 0.25]$ Additionally, four levels of t were tested, $t = [1, 10, 15, 20]$. These combinations were tested via an exhaustive grid search, while such thorough grid searches are intractable for large numbers of hyper-parameters (equiv. smaller sets of hyper-parameters over large ranges). In such situations it is advisable to instead use a method such as outlined in (Bihl *et al.*, 2020), or consider a commercial hyperparameter optimizer such as Optuna (Akiba *et al.*, 2019), Ray Tune (Lai *et al.*, 2018) or HyperOpt (Bergstra *et al.*, 2013).

The traditional CNN's overall accuracy on the test set was 74.97%, while the Bayesian augmented CNN's best-case overall accuracy was 75.17%. These similar accuracies are a

Table 2. Validation accuracy of tested optimizers

	Validation accuracy Traditional CNN	Bayesian CNN
Adam	98.65%	98.65%
NAdam	98.78%	98.02%
Adamax	98.82%	98.82%
Adagrad	96.79%	95.48%

Source(s): Authors' own work

positive indication, though the slight edge found with the Bayesian augmented CNN is well within the stochastic variability expected in training neural networks. Furthermore, our results indicate that α also has a minimal impact on aggregate accuracy (see Table 3).

The by-class accuracy is very similar for the standard CNN and the Bayesian CNN (with an $\alpha = 0.25$, making 20 predictions per frame). Figures 7 and 8 summarize these results. The core takeaway from these figures is that the shift from a standard frame-by-frame classification model (i.e. a traditional CNN) to a Bayesian weighted classification model comes with minimal downsides to both overall accuracy and class-by-class accuracy. Indeed, for some classes the Bayesian approach is better.

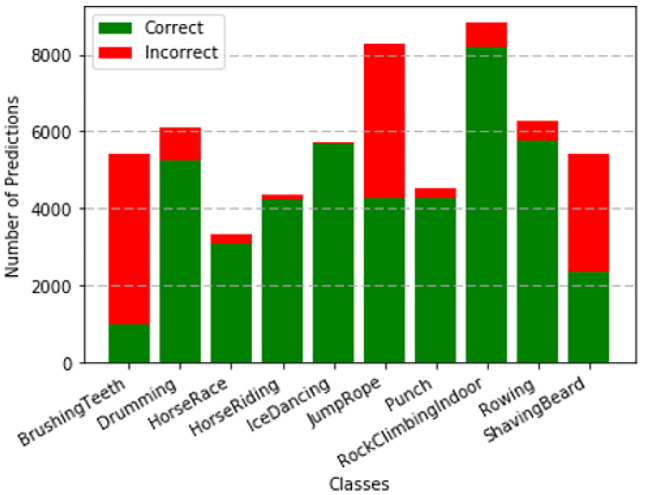
We also explored how the number of predictions per frame, *t*, impacted both accuracy and frames per second processed during classification. As seen in Figure 9, accuracy is quite insensitive to changes in *t* varying only 0.1% across all tested values. Conversely frame rate is quite sensitive to changes in *t*, decreasing approximately linearly.

Based on these hyperparameter tuning results we settled on $\alpha = 0.25$, with $t = 20$ predictions per frame. We defined a flickering metric (Equation 4), representing the total number of times the class prediction changes divided by the total number of frames. It is worth noting that this metric is symmetric and does not differentiate between class changes from the correct class to an incorrect class, versus the reverse.

Table 3. Overall accuracy for different α

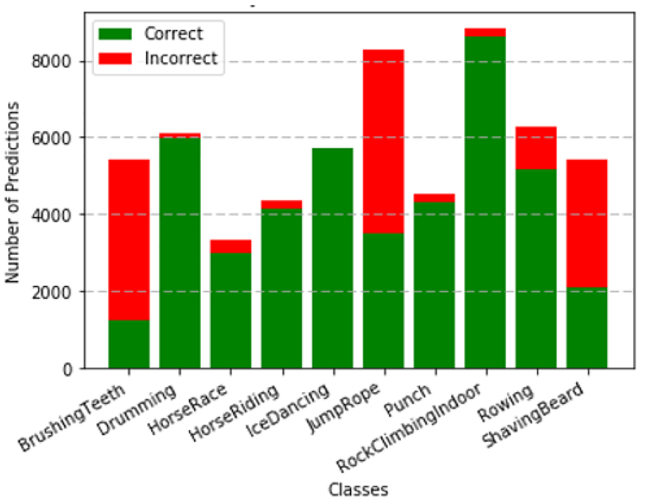
α	Accuracy
1.00	75.06%
0.75	75.10%
0.50	75.15%
0.25	75.17%

Source(s): Authors' own work



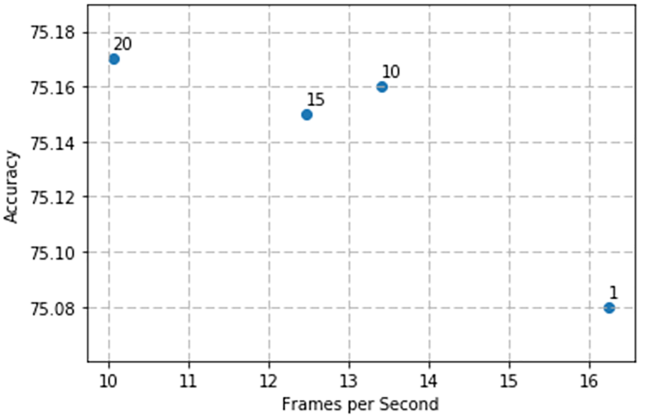
Source(s): Authors' own work

Figure 7. By-class classification accuracy for standard CNN



Source(s): Authors' own work

Figure 8. By-class classification accuracy for Bayesian CNN



Source(s): Authors' own work

Figure 9. Impact on frame-rate and accuracy based on number of classifications per frame, *t*

$$Flickering = \frac{Count\ of\ Class\ Changes}{Number\ of\ Frames} \tag{4}$$

At these hyperparameter settings the proposed Bayesian CNN model reduced flickering by 67.43% versus the traditional CNN. Another way to consider this decrease is that while on average the traditional CNN classifier changes class every 1.85 s, on average the Bayesian CNN changes class every 5.68 s. Considering the minimal (six frames per second) impact on framerate and the negligible differences in both overall accuracy and class-by-class accuracy this significant decrease in flickering indicates that the Bayesian CNN may be worth additional study.

4.2 CNN classification with smoothing

The approach from Section 3.3 was tested on a data set of 275 videos with 43,826 total frames. The CNN classification with smoothing used the ten video classes described in Section 3.1 with an 80/20 training/validation split. Equally weighted observations were used with $x = 1$ (i.e., no smoothing), 30 and 120 frames. The overall classification accuracy was similar in each of these cases (see Table 4).

Classification accuracy on the test set did not vary significantly with smoothing, although classification of individual videos differed. The number of misclassified videos using $x = 30$ vs $x = 120$ differed by one.

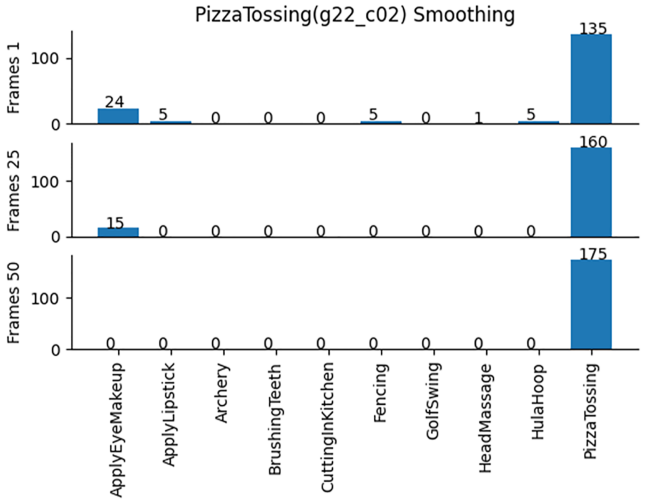
To understand the impact on classification flicker within a video clip, a frames classification analysis was conducted to see the effect of smoothing on intra-video classification. $x = 1, 25,$ and 50 were used for this portion of the analysis.

For videos producing multiple classifications, smoothing produced a marked improvement in classification flicker. Figure 10 shows representative results from a single video. For this video, each increased level of smoothing produces less classification flicker. With $x = 1$ (no smoothing), the frames are classified into five different classes, with 77.6% accuracy. However, with $x = 25$ equally weighted frames, accuracy improves to 92.0%, with only one frame misclassified. Finally, with $x = 50$ equally weighted frames, accuracy improves to 100.0% with no frames misclassified.

Table 4. Overall classification accuracy by number of frames averaged

x	Correct frames	Incorrect Frames	Accuracy
1	185	90	67.3%
30	185	90	67.3%
120	184	91	66.9%

Source(s): Authors' own work



Source(s): Authors' own work

Figure 10. By-class classification accuracy for pizza tossing video with varying x

To better illustrate this outcome, Figures 11–13 show individual frame classification (correct or incorrect) for the video clip considered in Figure 9. In Figures 11–13, $x = 1$, 25 or 50, respectively.

Figures 11 and 12 highlight classification accuracy for a frame in relation to neighboring frames. With no smoothing ($x = 1$, Figure 11), flickering is evident from the intermittent red bands. When red bands persist for more than two sequential frames, this can be attributed more to generally poor classification performance rather than to flickering. When $x = 25$ (Figure 12), all small flickering occurrences have been smoothed. Now instead of flickering, blocks of classifications are evident, where the classification changes for the final 15 frames. The inaccuracy in the final red block is attributed to the CNN front end being inadequate for a particular video action, as described below. Finally, with $x = 50$ (Figure 13), each frame is correctly classified.

Poor classification by the CNN front end can occur when the model is insufficiently sure of the classification and two (or more) classes are nearly equally likely. As the action depicted tends toward one class, the preponderance of the weighted average can shift toward the incorrect class leaving too few correct classifications in the weighted average to avoid a shift in classification.

Similar to the case shown in Figure 12, there were instances where smoothing could coerce an image classification to an incorrect classification depending on the relative uncertainty of the base CNN prediction of certain classes. In instances where the model is truly misclassifying the bulk of frames, then the smoothing technique will smooth correct classifications found intermittently within an incorrect classification to that incorrect classification.

5. Conclusions

This paper explored two methods to alleviate video classification flickering in near real-time for video footage. The pervasive phenomenon of video classification flickering is when occasional misclassifications occur frame-to-frame. One cause for flickering is that traditional video classification treats inputs as independent observations which ignores the temporally linked nature of video data. This work explored two methods to alleviate flickering: (1) smoothing of classifications, and (2) augmenting CNNs with Bayesian layers to add memory.

The augmentation of Bayesian layers to CNNs matched with an exponentially decaying weighted average for classifications demonstrated promising benefits in reducing flickering. In the best case the proposed Bayesian CNN model reduced flickering by 67% while

Source(s): Authors' own work

Figure 11. No smoothing ($x = 1$), sequential frames (Red indicates misclassification)

Source(s): Authors' own work

Figure 12. Frame smoothing ($x = 25$), sequential frames (Red indicates misclassification)

Source(s): Authors' own work

Figure 13. Frame smoothing ($x = 50$), sequential frames (Red indicates misclassification)

maintaining both overall accuracy and class level accuracy. The two downsides to the approach are that training the Bayesian CNN is more computationally demanding and the requirement to classify frames multiple times reduces resulting framerate. However, for some high surety mission applications this is a tradeoff the decision analyst may be willing to make.

Smoothing of frame predictions worked effectively, with the smoothing technique relying heavily on the base model predictions. Moreover, the technique works best in cases where flickering is intermittent. Consequently, the technique performs poorly when many misclassifications are made in sequence, which leads to smoothing away correct classifications.

Eliminating neural network misclassification is an issue at the leading edge of AI research, focusing effort on enhancing the capabilities of the underlying model. This research scopes this problem through the lens of video classification flickering, where misclassifications are the minority in an expanse of correct classifications. Therefore, we took a different approach of making additions to a CNN and pursuing smoothed results with a potentially low impact to the speed at which the model runs.

This research paves the way for additional research and poses a natural third research question:

RQ3. Can the combination of Bayesian augmentation and smoothing further reduce video-classification flickering?

Several intriguing directions exist for future work: The logical follow-ons to the research instituted within this manuscript entails implementing smoothing and/or Bayesian augmentation techniques, together, or individually, into a multi-object detection algorithm. Such an approach requires the development of an accounting method for each potential object in the frame, as well as a tracking method to create temporal linkages between frames. A thematically different approach which appears to show promise is the usage of recurrent neural networks within a video classification/objection detection framework to handle the temporal linkages of the data. Such an approach would build off work done in Lai *et al*. (2018) and Swize *et al*. (2022).

References

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Kayama, M. (2019), "Optuna: a next-generation hyperparameter optimization framework", *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Vol. 18, pp. 2623-2631, doi: 10.1145/3292500.3330701.

Andrejevic, M. (2019), "Automating surveillance", *Surveillance and Society*, Vol. 17 Nos 1/2, pp. 7-13, doi: 10.24908/ss.v17i1/2.12930.

Ball, J., Anderson, D. and Chan Sr, C. (2017), "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community", *Journal of Applied Remote Sensing*, Vol. 11 No. 4, 042609, doi: 10.1117/1.jrs.11.042609.

Bihl, T. and Talbert, M. (2020), "Analytics for autonomous C4ISR within e-government: a research agenda", *Proceedings of the 53rd Hawaii International Conference on System Sciences (HICCS)*, pp. 2218-2227.

Bergstra, J., Yamins, D. and Cox, D.D. (2013), "Hyperopt: a Python library for optimizing the hyperparameters of machine learning algorithms", *SciPy'13*.

Bihl, T., Schoenbeck, J., Steeneck, D. and Jordan, J. (2020), "Easy and efficient hyperparameter optimization to address some artificial intelligence 'ilities'", *Proceedings of the 53rd Hawaii International Conference on System Sciences*, pp. 943-952.

Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D. (2015), "Weight uncertainty in neural networks", *International Conference on Machine Learning*, pp. 1613-1622.

Bochkovskiy, A., Wang, C. and Liao, H. (2020), "YOLOv4: optimal speed and accuracy of object detection", *arXiv preprint*, arXiv:2004.10934.

Gal, Y. and Ghahramani, Z. (2016), "Dropout as a Bayesian approximation: representing model uncertainty in deep learning", *International Conference on Machine Learning*, pp. 1050-1059.

Géron, A. (2023), *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 3rd ed., O'Reilly Media, Sebastopol, CA.

Goan, E. and Fookes, C. (2020), "Bayesian neural networks: an introduction and survey", *Case Studies in Applied Bayesian Data Science*, Fall 2018, pp. 45-87, doi: 10.1007/978-3-030-42553-1_3.

Graves, A. (2011), "Practical variational inference for neural networks", *24th Conference on Advances in Neural Information Processing Systems*, Vol. 24.

He, K., Zhang, X., Ren, S. and Sun, J. (2016), "Deep residual learning for image recognition", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

Jain, A.K., Duin, R.P.W. and Mao, J. (2000), "Statistical pattern recognition: a review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22 No. 1, pp. 4-37, January, doi: 10.1109/34.824819.

Kandel, E., Siegelbaum, S., Mack, S. and Koester, J. (2021), *Principles of Neural Science*, 6th ed., McGraw-Hill Education/Medical, New York.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei-Fei, L. (2014), "Large-scale video classification with convolutional neural networks", *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725-1732.

Kullback, S. and Leibler, R. (1951), "On information and sufficiency", *The Annals of Mathematical Statistics*, Vol. 22 No. 1, pp. 79-86, doi: 10.1214/aoms/1177729694.

Kurian, E., Kizhakethottam, J. and Mathew, J. (2020), "Deep learning based surgical workflow recognition from laparoscopic videos", *5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 928-931, doi: 10.1109/icces48766.2020.9137855.

Lai, W.-S., Huang, J., Wang, O., Shechtman, E., Yumer, E. and Yang, M. (2018), "Learning blind video temporal consistency", *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 170-185.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, Vol. 86 No. 11, pp. 2278-2324, doi: 10.1109/5.726791.

McCloskey, B., Cox, B., Champagne, L. and Bihl, T. (2023), "Benefits of using blended generative adversarial network images to augment classification model training data sets", *Journal of Defense Modeling and Simulation*, doi: 10.1177/15485129231170225.

Office of the Secretary of Defense (OSD) (2022), *National Defense Strategy of the United States of America*, Washington DC.

Shah, R., Urmonov, O. and Kim, H. (2021), "Improving performance of CNN based vehicle detection and tracking by median algorithm", *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Vol. 7, pp. 1-3, doi: 10.1109/icce-asia53811.2021.9641942.

Simonyan, K. and Zisserman, A. (2015), "Very deep convolutional networks for large-scale image recognition", *arXiv preprint*, arXiv:1409.1556.

Soomro, K., Zamir, A. and Shah, M. (2012), "UCF101: a dataset of 101 human actions classes from videos in the wild", *arXiv preprint*, arXiv:1212.0402.

Stamoulakatos, A., Cardona, J., Michie, C., Andonovic, I., Lazaridis, P., Bellekens, X., Atkinson, R., Hossain, M. and Tachtatzis, C. (2021), "A comparison of the performance of 2D and 3D convolutional neural networks for subsea survey video classification", *Oceans 2021: San Diego-Porto*, IEEE, pp. 1-10.

Sweatt, J.D. (2016), "Neural plasticity and behavior—sixty years of conceptual advances", *Journal of Neurochemistry*, Vol. 139 No. 2, pp. 179-199, doi: 10.1111/jnc.13580.

Swize, E., Champagne, L., Cox, B. and Bihl, T. (2022), "Bayesian augmentation of deep learning to improve video classification", *2022 Hawaii International Conference on System Sciences*, pp. 2097-2106.

Tishby, N., Levin, E. and Solla, S. (1989), "Consistent inference of probabilities in layered networks: predictions and generalization", *International 1989 Joint Conference on Neural Networks*, IEEE, pp. 403-409.

Wen, Y., Vicol, P., Ba, J., Tran, D. and Grosse, R. (2018), "Flipout: efficient pseudo-independent weight perturbations on mini-batches", *arXiv preprint*, arXiv:1803.04386.

Xu, K., Wen, L., Li, G., Qi, H., Bo, L. and Huang, Q. (2021), "Learning self-supervised space-time cnn for fast video style transfer", *IEEE Transactions on Image Processing*, Vol. 30, pp. 2501-2512.

Corresponding author
Lance Champagne can be contacted at: lance.champagne@afit.edu