AI-Enhanced Lighting Activation for Awakening Passengers

Kim, Dayoung
Stanford University
dkim9613@stanford.edu

Song, Wanbin
Stanford University
wanbins@stanford.edu

Abstract

This paper proposes a deep learning-based system for intelligent lighting activation in aircraft cabins, aiming to gently awaken passengers at optimal times by recognizing their activity patterns. We utilized a publicly available action recognition repository from IBM as a foundation and integrated advanced components to improve accuracy and efficiency. Our method begins with video input captured by cameras installed above passenger seats, which are processed using a ResNet18 backbone for spatial feature extraction. The Temporal Action Module (TAM) then captures temporal dependencies across frames. To enhance performance, we incorporated TokenLearner and TokenFuser modules, to efficiently process and integrate spatial and temporal features. We trained our model using a cross-entropy loss function and evaluated it on a prepared dataset, achieving significant improvements in classification accuracy. Our results demonstrate the effectiveness of combining traditional CNN-RNN architectures with cutting-edge token-based approaches, providing a robust solution for intelligent lighting activation in aircraft cabins. This system not only enhances passenger comfort but also represents a state-of-the-art approach in the application of deep learning to smart cabin systems.

1. Introduction

In recent years, the airline industry has prioritized enhancing passenger experiences during flights. A crucial aspect of this enhancement is optimizing the wake-up process to minimize jet lag and align with meal services. To address this, we propose the development of a deep learning-based system for intelligent lighting activation in aircraft cabins. This system aims to gently awaken passengers at optimal times by recognizing their activity patterns.

Our motivation for pursuing this problem is from our background in the aviation industry and our interest in improving passenger experiences and smart cabin systems. We believe that an intelligent wake-up system could significantly enhance the quality of service offered by airlines, differentiating them in a competitive market. The input to our algorithm is a streaming video captured by cameras installed above passenger seats. We utilize a deep learning model to process this video input. The output of our system is a notification to the flight crew indicating whether a passenger is awake. This ensures timely wake-ups and synchronization with meal services, contributing to a more comfortable and efficient travel experience for passengers.

In our baseline module development, we have demonstrated the effectiveness of this approach by leveraging the spatial feature extraction capabilities of 2D Convolutional Neural Networks (CNNs). These networks are adept at identifying intricate patterns within individual frames. To capture the temporal dependencies essential for accurate activity recognition, we integrated temporal modeling components, which significantly enhance the system's performance

Furthermore, to improve the overall accuracy and reliability of our system, we incorporated recent advancements in token-based approaches for recognition tasks. This integration allows our system to process and interpret video data more efficiently, ensuring a higher quality of service for airline passengers. By combining state-of-the-art techniques in deep learning and activity recognition, our intelligent lighting activation system represents a significant advancement in enhancing passenger experiences during flights.

2. Related Work

The development of intelligent systems for passenger comfort in aircraft has been explored in various studies. Existing research can be categorized into three main approaches: sensor-based systems, camera-based systems, and hybrid systems.

2.1. Sensor-based Systems

Sensor-based systems often utilize wearable devices to monitor physiological parameters such as heart rate, body temperature, and movement. For instance, [1] developed a wearable sensor system to detect sleep patterns and stress levels during flights. These systems are highly accurate in

physiological measurement but suffer from inconvenience and discomfort for passengers who need to wear devices throughout the flight.

2.2. Camera-based Systems

Camera-based systems leverage image and video processing to monitor passenger states. Many of researches focues on driver's fatigue recognition using facial features from image from camera[2]. While these systems are non-intrusive, they often struggle with varying lighting conditions and occlusions, impacting accuracy. Additionally, the study by Zheng et al. [3] on real-time detection of driver fatigue using a CNN-LSTM model showcases the potential of combining convolutional layers for feature extraction with LSTM layers for temporal sequence modeling. This method provides improved accuracy in detecting fatigue states but requires significant computational resources. These approaches are clever in their use of powerful image processing capabilities but still face challenges with dynamic environments.

2.3. Summary of Strengths and Weaknesses

Overall, sensor-based systems provide precise physiological data but lack convenience. Camera-based systems offer non-intrusive monitoring but face environmental challenges. Our proposed system aims to balance these trade-offs by leveraging the strengths of camera-based monitoring with advanced deep learning techniques, providing a robust and user-friendly solution for intelligent lighting activation in aircraft cabins.

2.4. Video understanding

If we only focus on video understanding problem, there are lots of related researches and progress with the introduction of a number of large-scale video datasets, such as Kinetics, Moments-In-Time and YouTube-8M. Recent models have highlighted the importance of efficiently modeling spatiotemporal information for video action recognition. Many of successful deep learning architectures for action recognition are usually based on two-stream model [4], which process RGB frames and optical flow in two seperate CNNs with a late fusion in upper layers [5]. Two stream approaches have been utilized in a range of action recognition methods [[6], [7], [8], [9]]. Another straightforward yet popular method involves using 2D CNNs to extract frame level features and then modeling the temporal casuality. For example, TAM[10] is based on depthwise 1x1 convolutions to capture temporal dependencies across frames efficiently. Another approach involves using 3D CNNs, which build upon the success of 2D models in image recognition[11] to identify actions in videos. For example, C3D[12] utilizes 3D ConvNets, outperforming 2D CNNs by leveraging large-scale video datasets.

3. Methods

In our project, we aimed to develop a system for intelligent lighting activation in aircraft cabins based on passenger activity recognition. To achieve this, we utilized deep learning algorithms to analyze video input from cameras installed above passenger seats. We started with a publicly available repository from IBM, specifically designed for action recognition using PyTorch [13]. This repository provided a strong foundation for our work, allowing us to focus on refining the model and adapting it to our specific use case.

The input to our algorithm is streaming video captured by cameras. These videos are processed frame by frame to extract relevant features. Each frame is represented as a tensor of pixel values, typically of shape (C, H, W) where C is the number of color channels, H is the height, and W is the width. Our system processes these frames to predict the activity state of passengers, particularly focusing on identifying if a passenger is awake or asleep. The output of our algorithm is a binary classification indicating whether a passenger is awake (1) or asleep (0). This output is used to control the cabin lighting system and notify flight crews accordingly.

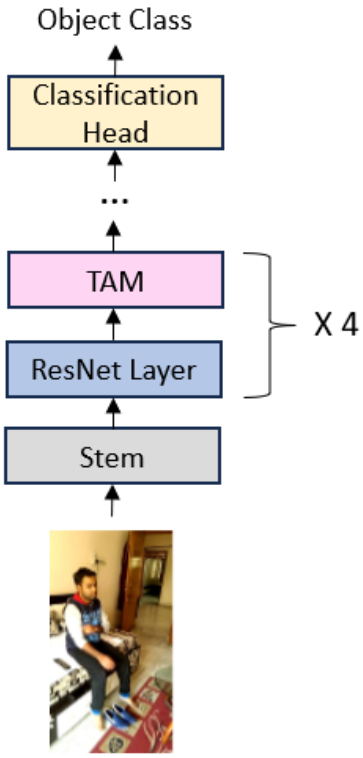

Figure 1. Baseline Model Architecture

For our baseline model, we utilized ResNet18 as the

backbone for feature extraction and a Temporal Action Module (TAM) for capturing temporal dependencies in video sequences. ResNet18 is a widely used Convolutional Neural Network (CNN) that efficiently extracts spatial features from input frames. The key component of ResNet18 is the residual block, which helps in training deep networks by mitigating the vanishing gradient problem. In order to learn temporal dependencies in video sequences, our baseline approach[13] integrates Temporal Adaptive Modules (TAM)[10] and temporal pooling layers, extending the ResNet-18 architecture. Each BasicBlock has a TAM included within it which processes features from the previous, current, and next frame using Squeeze-and-Excitation (SE) modules, enabling adaptive focus on important temporal features. After certain residual layers, the temporal pooling layer are added to gradually downsample the temporal dimension while keeping important temporal information. Then the input's journey throughout the architecture would be, first, convolutional and pooling layers process the input tensor, which has been reconfigured to handle each frame separately. TAM-enhanced residual blocks come next. Global average pooling and dropout are applied after the last residual layer, and the tensor is then passed through a fully connected layer for classification. The output is then reshaped and averaged across frames to produce the final prediction, effectively leveraging temporal context for improved action recognition in videos.

To improve the baseline accuracy, we incorporated two additional components: TokenLearner[14] and TokenFuser. These components are inspired by the Vision Transformers (ViTs) architecture and help in efficiently capturing both spatial and temporal dependencies.

The effect of the implementation of TokenLearner and TokenFuser is that the attention module can learn which tokens would be used for the recognition task. It uses a set of selected tokens instead of all the fixed tokenized inputs. With this adaptive tokenization and dynamic selection, overall process becomes efficient.

TokenLearner selects the informative combination of pixels using a weight map. The spatial attention module for TokenLearner and TokenFuser uses 3 convolutional layers with ReLU with the last layer generating attentions with sigmoid. These spatial attention modules in TokenLearner learn which regions seem to be more important and generate tokens based on their adaptive learning. With the created tokens, they compute a weight map from the tokens and multiply the input to the weights. Then, the global average pooling layer reduces the dimensionality to have tokens as dense as the chosen size, 8 or 16 in this project. Figure 1 shows TokenLearner applying each spatial attention to the subsets of input tensor and generating tokens. The formula of calculating token z is in equation (1), where X is an input, A is an intermediate weight tensor, α is a function that

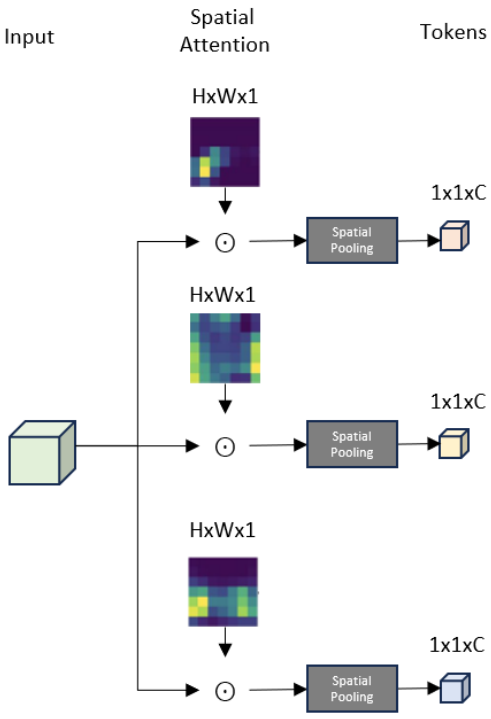

Figure 2. TokenLearner

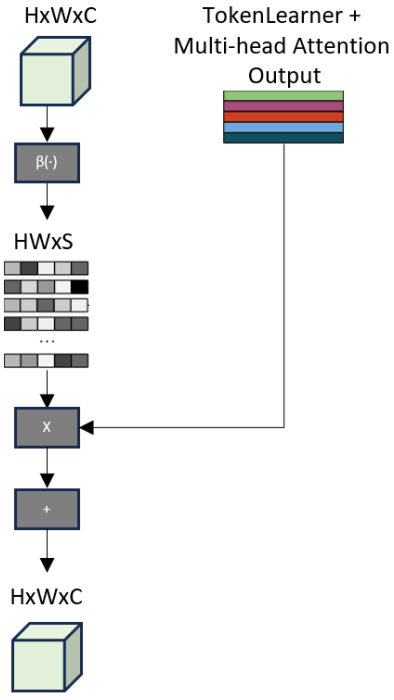

Figure 3. TokenFuser

calculates A, γ is the broadcasting function, and ρ is a spatial global average pooling function.

$$Z_i = A_i(X_i) = \rho(X_t \bigodot A_{iw}) = \rho(X_t \bigodot \gamma(\alpha_i(X_t))) \quad (1)$$

The generated tokens are then fed into a multi-head attention layer. TokenFuser utilizes the learned information, the output of multi-head attention module, to remap them to the original input tensor generating the same size output with the input of TokenLearner. Figure 2 shows how the output from TokenLearner and multi-head attention and the original input tensor are computed inside of the TokenFuser module. The equation (2) explains computation within TokenFuser, where each output of TokenLeaner and multi-head attention is Y_t, the input tensor is X_t,and the intermediate weight tensor is B_w computed by the function β.

$$X_t^{j+1} = B(Y_t, X_t^j) = B_w Y_t + X_t^j = \beta_i(X_t^j) Y_t + X_t^j \quad (2)$$

Having the input size of TokenLearner and the output size of TokenFuser same, it makes easier to apply this set of TokenLearner and TokenFuser after any convolutional layer in ResNet-18 architecture.

4. Dataset

The dataset utilized in this project is the Charades dataset[15], a publicly available collection of videos with various household activities. The Charades dataset is designed to facilitate research in activity recognition, offering annotated video clips that capture a wide range of actions performed in indoor environments. It has 157 activity classes with 8000 training and 1686 validation videos of 30 seconds on average. However, our project focuses on a subset of the Charades dataset, specifically targeting the actions "Someone is awakening in bed" (action ID: c133) and "Lying on a bed" (action ID: c134). These actions were chosen due to their relevance to our project objectives, and each action was mapped to a numeric label to facilitate processing, to label 0 for 'c133', and 1 for 'c134'. With these, we've implemented data pre-processing module which includes loading metadata from a given CSV file, filtering actions, extracting frames from video files, and writing the metadata text file to be used during the training stage. The detailed processed dataset statistics are shown in Table 1.

Dataset	Total Videos	Total Frames	c133	c134
Train	257	8,635	102	155
Val	82	2,154	45	37

Table 1. Summary of the Processed Charades Dataset

Focusing on specific actions and employing systematic frame extraction and data preparation techniques, this project constructs a robust dataset tailored to the research objectives. The combination of detailed metadata, organized frame storage, and clear labeling facilitates effective

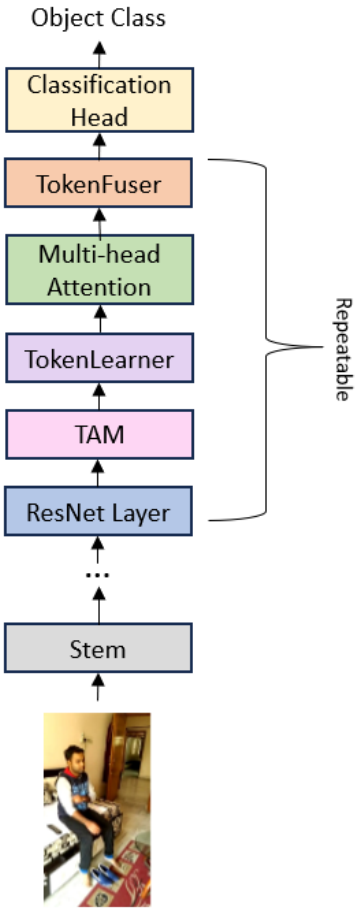

Figure 4. Model Architecture with TokenLearner, Multi-head Attention, and TokenFuser

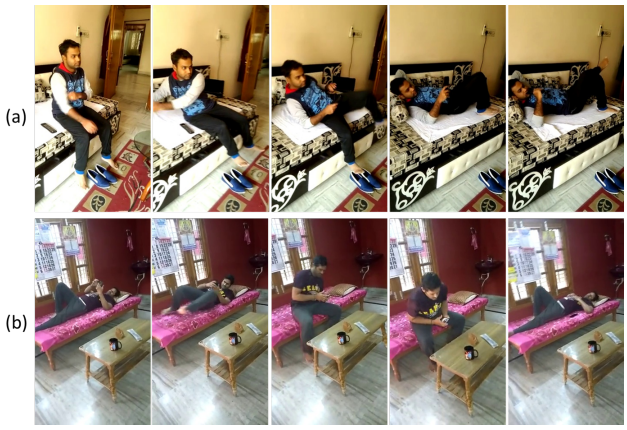

Figure 5. Examples of dataset. (a) c133 (b) c134

analysis and model development.

5. Experiments/Results/Discussion

We utilized the ResNet-18 backbone with the TAM and the model was trained for 100 epochs with a SGD optimizer, learning rate of 0.001, a batch size of 16. We chose learning rate to be 0.001 because it showed better performance than 0.05 and others. The SGD optimizer is chosen because of its fastness. Also, it's known to give a good generalization.

Model	**Loss** (Train/Val)	**Accuracy** (Train/Val)
ResNet-18	0.613 / 0.813	62.868% / 48.718%
ResNet-18 w. TL & TF after L1, T = 8	0.341 / 1.017	84.191% / 48.718%
ResNet-18 w. TL & TF after L3, T = 16	0.303 / 1.028	87.500% / 60.256%
ResNet-18 w. TL & TF after L3,4, T = 16	0.418 / 0.780	79.779% / 61.539%

Table 2. Training and Validation Performance of Models

There are three experiments that have significant improvement we found to explain about in Table 2. The first experiment with the set of TokenLearner and TokenFuser was to apply them after the first layer of ResNet-18. With 8 tokens used, the training performance significantly improved from the baseline model. However, the validation accuracy remained same with higher loss. One of the next experiments to improve the model was applying the set after the third layer with 16 tokens. In this experiment, both training and validation accuracy was higher than the base line and the first experiment. The fact that the validation loss was higher could mean that when the model misclassifies the data, it makes a big error. The final experiment was applying the set after the third layer and the fourth layer with 16 tokens. Although we expected both train and validation accuracy to be higher than previous ones, only the validation accuracy improved a little and the loss and the accuracy of training was a little less.

As you can see in Figure 4, The confusion matrices for the training and validation datasets show that the model correctly classified 82 out of 102 samples for class c133 and 123 out of 155 samples for class c134 in the training set. For the validation set, the model correctly classified 27 out of 45 samples for class c133 and 23 out of 37 samples for class c134, indicating a higher performance on the training data compared to the validation data. As mentioned in the paper [14], applying the set of TokenLearner and TokenFuser after 3/4 of the model structure showed better performance than

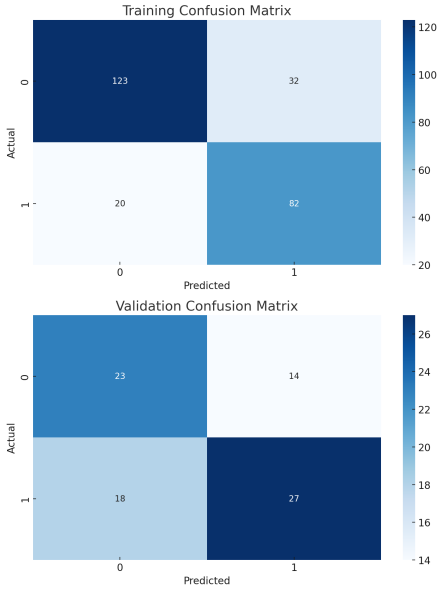

Figure 6. Confusion matrices for t

applying it close to the input layer. Also, repeating the set to multiple layers helped generalizing the model. In addition, the number of tokens also seemed to be affecting the performance. When we used more tokens without any change in other settings, there were 1 or 2% improvement in accuracy scores. Although the paper [14] mentions about computational expensiveness with more tokens, but the time difference was not much and was able to be compensated for the better performance. Therefore, among the experiments, the last model, ResNet-18 with the set of TokenLearner and TokenFuser after third and fourth layer with 16 tokens, could be considered as the best considering both train and validation performance indicating the most generalized model.

6. Conclusions/Future Work

The experiments of this project showed the effect of TokenLearner and TokenFuser. Learning representations of input tensor and adding a multi-head attention module improved the performance of classification problem. Also, how applying them to the model was important to make more generalized model, being able to make good predictions on new data. Over the experiments, although we were able to improve the performance and decrease the gap between the train and validation accuracy and loss, more ways could exist to enhance the model. The possible improvements include modifications in backbone model and modifications in the set of TokenLearner and TokenFuser.

The ResNet-18 backbone we used has 4 convolutional layers. If experiments with more layers in the model or with models with different architecture were conducted, we might have seen difference in performance scores. If the

architecture is different, the set of TokenLearner and TokenFuser might have been needed to applied differently and their learning mechanism also could have been changed, affecting the performance.

Another possible modification that could improve the performance is changing the spatial attention module. The spatial attention module we used uses three convolution layers. We chose to use three layers because when We worked with one convolutional layer, it advanced overfitting. There could be other architectures of spatial attention that improves the performance and reduces overfitting.

The other hyperparameter settings could exist, which best fits our dataset. For example, the initial number of tokens were chosen based on the paper [14]. However, since the dataset is different, our dataset might advantage from more tokens or less. With more time and resources, the experiments with the number of tokens and other hyperparmaeter tuning could improve the result.

Since the main problem statement of this project is to recognize flight passenger's awakeness, the dataset of people sitting in flight would have been more useful. Since we weren't able to get the data of flight passengers and their actions, it could be one of future limitations when implementing the system in the real world.

7. Contributions Acknowledgements

D.K designed and implemented the set of TokenLearner and TokenFuser with the reference of a public repository, `https://github.com/ariG23498/TokenLearner/tree/master`, and wrote the paper. W.S designed and implemented data preprocessing logic, baseline architecture with the reference of a public repository, `https://github.com/IBM/action-recognition-pytorch`, and wrote the paper.

References

[1] Akane Sano and Rosalind W Picard. Stress recognition using wearable sensors and mobile phones. In *2013 Humaine association conference on affective computing and intelligent interaction*, pages 671–676. IEEE, 2013.

[2] Zuopeng Zhao, Nana Zhou, Lan Zhang, Hualin Yan, Yi Xu, and Zhongxin Zhang. Driver fatigue detection based on convolutional neural networks using em-cnn. *Computational intelligence and neuroscience*, 2020, 2020.

[3] Ming-Zhou Liu, Xin Xu, Jing Hu, and Qian-Nan Jiang. Real time detection of driver fatigue based on cnn-lstm. *IET Image Processing*, 16(2):576–595, 2022.

[4] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

[5] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[6] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226, 2015.

[7] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[8] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 759–768, 2015.

[9] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[10] Quanfu Fan, Chun-Fu Chen, Hilde Kuehne, Marco Pistoia, and David D. Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. *CoRR*, abs/1912.00869, 2019.

[11] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[12] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[13] Chun-Fu Chen, Rameswar Panda, Kandan Ramakrishnan, Rogerio Feris, John Cohn, Aude Oliva, and Quanfu Fan. Deep analysis of cnn-based spatio-temporal representations for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[14] Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *CoRR*, 2022.

[15] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. *CoRR*, abs/1604.01753, 2016.