Human Action Detection

Final Project Report

ECS 170 Spring 2024

Yi-Hsuan Chiang[1], Jonathan Huang[1], Kate Xie[1], James Xu[1], and Xupeng Zhang[1]

[1]Department of Computer Science, University of California, Davis

June 12, 2024

1 Introduction

In the rapidly evolving field of artificial intelligence and computer vision, the development of accurate and reliable models for purposes such as human action detection has become more viable. In order to achieve human action detection, it can be fundamental for numerous other image detection, therefore paving the road for advancements in future computer vision technology. Our project focuses on human activity prediction, and our goal is to develop a comprehensive model that can accurately predict human actions from the given picture using machine learning techniques. Our project is positioned within the broader scope of computer vision, leveraging advanced machine learning algorithms to interpret and understand visual inputs.

2 Background

Human action detection involves two main steps: automatic recognition and classification of human actions. This technology has been widely applied in various fields, including surveillance, sports analytics, healthcare, and human-computer interaction. The ability to automatically detect and classify human actions enables enhanced security monitoring, improved athletic performance analysis, better patient care, and more intuitive user interfaces. However, several challenges hinder the effectiveness of human action detection. These challenges include the variability in how different individuals perform actions, the presence of complex or cluttered backgrounds, and the difficulty of accurately interpreting images where the human body is partially obscured.

Historically, human action detection relied on traditional machine learning algorithms and handcrafted features, such as Histograms of Oriented Gradients (HOG) and Optical Flow. However, these methods often struggled with the variability and complexity inherent in real-world datasets. The limitations of these traditional approaches have been addressed with the advent of advanced machine learning and deep learning techniques. Convolutional Neural Networks (CNNs) excel at recognizing patterns and features in images, making them well-suited for the task of human action detection. By leveraging the power of CNNs, modern human action detection systems can more accurately and efficiently identify and classify a wide range of human activities, even in challenging conditions. This progress highlights the significant strides made in the field and the potential for even greater advancements in the future.

3 Methodology

3.1 Dataset

To train and evaluate our human action detection model, we will be using a comprehensive dataset sourced from Kaggle created by Emirhan AI. The dataset contains a diverse collection of images labeled with various human activities such as walking, running, jumping, and more complex actions like dancing and playing sports. Each image is meticulously labeled to facilitate accurate model training and evaluation. Since we will be focusing on supervised machine learning, this dataset fully fulfills our requirements. The extensive range of scenarios captured in the dataset, including different environments, lighting conditions, and viewpoints, will challenge our model to adapt and perform reliably across varied real-world settings. Given the large amount of data contained in the dataset, efficient data handling and processing techniques will be essential. The only issue with this dataset is that it requires lots of computational power since the dataset is tremendous. However, our team has access to computers that are able to run models on large datasets like this one. We will take advantage of the dataset and the computing equipment aiming to develop a highly accurate and versatile human action detection model that can be applied in real-world applications.

3.2 Tools and Libraries

The following Machine Learning libraries were used:

1. os: Imported as an operating system to manage file paths and directories
2. numpy: Imported as np to do numerical and array operations
3. pandas: Imported to do data manipulation (reading the CSV file)
4. matplotlib: Imported to plot and visualize the data
5. seaborn: Imported to supplement matplotlib to visualize data and create statistical graphics
6. scikit-learn: Used for various machine learning tasks such as model evaluation and preprocessing
7. tensorflow: Imported as tf to build and train our neural network

8. keras: Used in conjunction with TensorFlow for building and training neural networks
9. pytorch: Used for implementing and experimenting with advanced neural network architectures like EfficientNet
10. timm: A PyTorch library for implementing EfficientNet and other advanced vision models
11. Flask: Used to build our web application and APIs

3.3 Data Processing

The dataset we find has been labeled, however, in different sizes. For our convenience, we resize to 128x128 pixels before any further processing. We decided to split the dataset in a 5:1 training-testing ratio for our training purposes. As a result, for each of the 15 different action categories, there are 1000 images in the training dataset and 200 images in the testing dataset. All testing images are batched into groups of 128, and each output label is encoded into a one-hot categorical format. Categories include calling, clapping, cycling, dancing, drinking, eating, fighting, hugging, laughing, listening to music, running, sitting, sleeping, texting, and using a laptop.

After the preparation, we further processed our image data using ImageDataGenerator() from TensorFlow, in which we specified the parameters in the generator, such as setting rotation functions, and the image quality parameters such as rescaling and filling mode after translation. We then input our splitting training and testing dataset into these generators, letting them take care of the image format and ensure the image validation. After doing so, all 18000 images were normalized and standardized and were ready to use in various models.

3.4 Selection Among Various Models

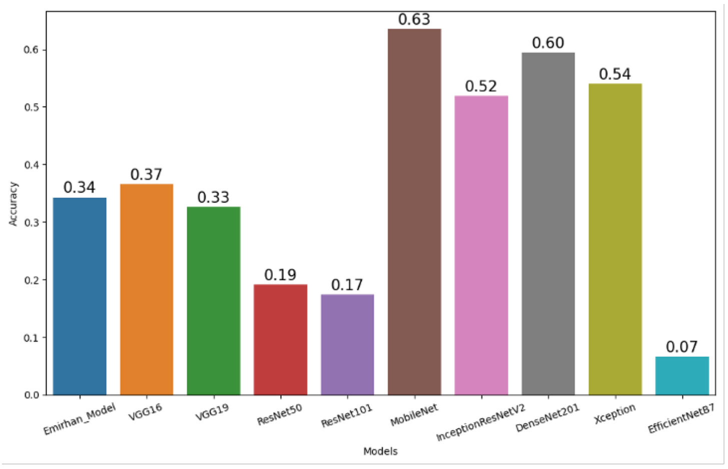


Figure 1: Validation Accuracy of Pre-trained Models drawn from Kaggle Dataset

We were initially inspired by the pre-trained models used on the Kaggle Dataset, where a various range of models are proposed and trained: VGG16, VGG19, Emirhan_model constructed by the author Emirhan, ResNet50, MobileNet, DenseNet201, and Xception, etc. Among these models, We observed the drastic differences in validation accuracy and computational efficiency: the accuracy ranged from as low as 7% up to a high of 63%. We recorded and used the comparatively best model among them, in which we finally selected the MobileNet, the highest accuracy model reaching 63% while being the most efficient (2-3 minutes to train). Denote that other models like VGG and ResNet are more computationally expensive (10-25 minutes to train) but still achieve a lower accuracy, and thus we omitted them. Therefore, we decided to go further with the MobileNet CNN model and potentially increase its accuracy.

3.5 Model Development

MobileNet is a lightweight deep neural network known for its efficiency in terms of computational usage. In addition to its 28 base layers, we added 4 custom layers to our model. One is an average pooling layer to reduce spatial dimensions, two dense layers with 64 neurons each using the ReLU activation function, and one output dense layer using the softmax activation function to output probabilities for each category. The model is compiled with categorical_crossentropy and Adam optimizer, with early stopping to monitor validation loss and prevent overfitting. The MobileNet model plateaus around 65% validation accuracy, so we wanted to explore another model, EfficientNet, to further improve our final accuracy.

EfficientNet is an advanced deep neural network recognized for its balanced scaling method. The training process involves more aggressive data augmentation and advanced regularization techniques to improve generalization. For our project, we utilized the EfficientNetV2-S model, which was pretrained and fine-tuned for our specific classification task. The model was trained using the CrossEntropyLoss function and optimized with the SGD optimizer. A learning rate scheduler was employed to adjust the learning rate dynamically. In addition, we added a dropout rate of 0.3 to motivate faster convergence and better generalization. The training process involved 10 epochs, with careful monitoring of training and validation loss and accuracy. Early stopping was implemented to prevent overfitting by monitoring the validation loss. The best model weights were saved based on the highest validation accuracy achieved during the training process.

After finalizing our EfficientNet Model, we built a basic web application using Flask where users can input images and predict human actions using the final model.

4 Results

4.1 Model Evaluation

To evaluate our results, we compared the final **validation accuracy** of our MobileNet and EfficientNet models. The MobileNet model is trained with 5 epochs, achieving a validation accuracy of 64%, whereas the EfficientNet model is trained with 10 epochs and achieved a staggering validation accuracy of 82%.

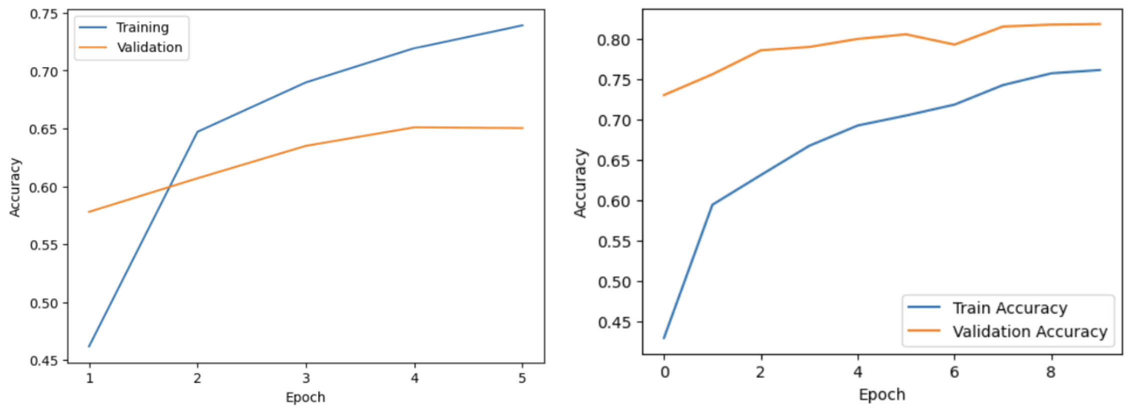
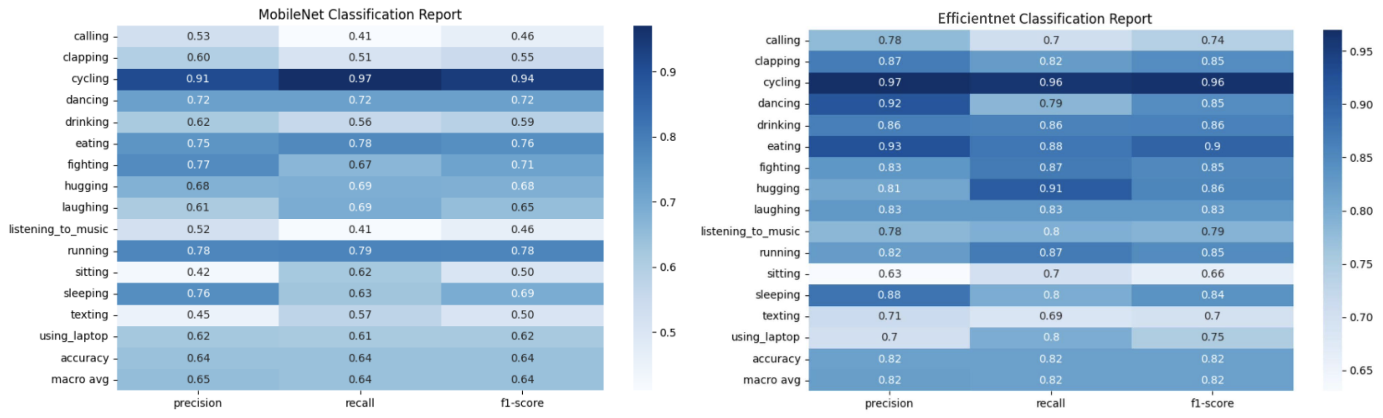


Figure 2: Training and Validation Accuracy at each Epoch (MobileNet VS EfficientNet)

One interesting trend we noticed for the EfficientNet model is that the validation accuracy surpasses the training accuracy even after 1 epoch, indicating that the model generalizes well on unseen data due to its aggressive data augmentation and regularization techniques. The MobileNet follows an upward trend of validation accuracy plateauing around 65%, and the training accuracy is consistently higher after 2 epochs.

We also evaluated our models based on **precision, recall, and f1 score** for each of the action categories. A trend we noticed is that actions involving objects such as calling, listening_to_music, sitting, and texting, generally have lower performance, likely due to the difficulties in recognizing object patterns. Overall, EfficientNet surpasses MobileNet in every aspect category for our classification task.



MobileNet Classification Report

	precision	recall	f1-score
calling	0.53	0.41	0.46
clapping	0.60	0.51	0.55
cycling	0.91	0.97	0.94
dancing	0.72	0.72	0.72
drinking	0.62	0.56	0.59
eating	0.75	0.78	0.76
fighting	0.77	0.67	0.71
hugging	0.68	0.69	0.68
laughing	0.61	0.69	0.65
listening_to_music	0.52	0.41	0.46
running	0.78	0.79	0.78
sitting	0.42	0.62	0.50
sleeping	0.76	0.63	0.69
texting	0.45	0.57	0.50
using_laptop	0.62	0.61	0.62
accuracy	0.64	0.64	0.64
macro avg	0.65	0.64	0.64

Efficientnet Classification Report

	precision	recall	f1-score
calling	0.78	0.7	0.74
clapping	0.87	0.82	0.85
cycling	0.97	0.96	0.96
dancing	0.92	0.79	0.85
drinking	0.86	0.86	0.86
eating	0.93	0.88	0.9
fighting	0.83	0.87	0.85
hugging	0.81	0.91	0.86
laughing	0.83	0.83	0.83
listening_to_music	0.78	0.8	0.79
running	0.82	0.87	0.85
sitting	0.63	0.7	0.66
sleeping	0.88	0.8	0.84
texting	0.71	0.69	0.7
using_laptop	0.7	0.8	0.75
accuracy	0.82	0.82	0.82
macro avg	0.82	0.82	0.82

Figure 3: Classification Report (MobileNet VS EfficientNet)

4.2 Misclassification Analysis

Even though the EfficientNet model does well in classifying actions, the misclassification can still happen. We selected three representative misclassifications by our model:

- First Figure: The model predicted using a laptop, but the actual class is calling.

- Second Figure: The model predicted calling, but the actual class is laughing.
- Third Figure: The model predicted fighting, but the actual class is hugging.

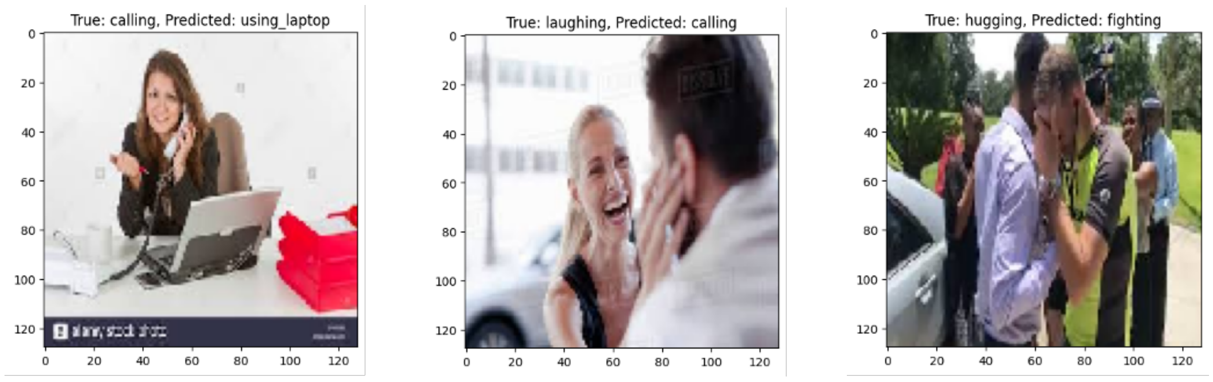

True: calling, Predicted: using_laptop
True: laughing, Predicted: calling
True: hugging, Predicted: fighting

Figure 4: Representative Image Misclassifications Conduct By EfficientNet

It is examinable that there are high correlation between the predicted class and the actual class due to their similarity: there is a laptop in front of the who is calling with her phone in first figure, A man is talking to a woman who is laughing with his hand on the ear in second figure, and two men are physically touching each other while there are hugging in third figure. However, note that there is no clear reason for the misclassification to happen: we cannot assume what, as human beings, we see is what the model developed and weighted on, such that the assumed correlation may not be true for the model to improve and specify on. Thus, the optimization of EfficientNet can be quite challenging.

5 Discussion

Throughout this project, we encountered various challenges and learning opportunities. These challenges and learning opportunities greatly influenced our approach and final results, and also provided us with a deep understanding of the fields of computer vision and machine learning. The variability in the behavioral performance of different individuals in the data we trained on and the complexity of the real-world environment tested the robustness and adaptability of our model. We initially chose the MobileNet model because of its high training efficiency and moderate accuracy. However, as we tested the limits of the model's performance, it became clear that we needed further improvements.

The decision to further train the EfficientNet model was a critical one. It significantly improved our model's accuracy on the validation set to 82%. This shift not only demonstrated the potential of using more complex architectures for classification tasks, but also highlighted the importance of choosing the right tools for the specific challenges of human action recognition. From this experience, we learned that achieving higher accuracy in complex action detection scenarios requires not only powerful models, but also a deep understanding of the nuances in the data.

6 Conclusion

Our project has laid a solid foundation for the development of advanced human action recognition systems. By using machine learning algorithms and a comprehensive and large dataset, we developed different models. And these models not only met but also exceeded our initial expectations in terms of accuracy and efficiency. The successful integration of EfficientNet has brought significant improvements to our system. This also highlights the potential for future enhancements and applications of our technology.

As we move forward, our focus will be on improving our model. Making our model more effective at handling the subtle differences between similar trades and exploring other models that may offer further improvements. We aim to improve our approach by addressing challenges discovered during testing, enhancing the model's ability to generalize across different scenarios, and exploring the application of our research that has a positive impact on society.

7 Contribution

7.1 Yi-Hsuan Chiang

Conducted the presentation slides and presented the sections on the dataset and data preparation. Contributed to the discussion and conclusion sections of the final report and formatted the entire document in LaTeX.

7.2 Jonathan Huang

Conducted the presentation, and examined and corrected its template and flow. Wrote the methodology portion of the final report, and examined the overall paper correctness.

7.3 Kate Xie

Enhance the model's accuracy. Create a template for the slides and contribute to the introduction, background, and motivation sections of both the presentation and the report.

7.4 James Xu

Experiment with the initial MobileNet model and build a simple web application demo for the final model. Contributed to the Methodology and Results portion of the project report and presentation.

7.5 Xupeng Zhang

Model training to improve the accuracy of the model. Look into GPU resources for faster training. As well as present the conclusion section of the presentation.

8 GitHub Repository

Here is the link for the GitHub that includes all the code for this project:
ECS170_FinalProject_5

9 Referenced Notebook

Human-action-detection-using-cnn

10 Citations

Alfaifi, R., Artoli, A.M. Human Action Prediction with 3D-CNN. SN COMPUT. SCI. 1, 286 (2020).