Human Activity Recognition using Deep learning

Gagan Deep Singh
Western University
London, Canada
gsing363@uwo.ca

Gurjot Singh
Western University
London, Canada
gsing352@uwo.ca

Mansimar Singh Bhatia
Western University
London, Canada
mbhati28@uwo.ca

Sherry Ahuja
Western University
London, Canada
sahuja49@uwo.ca

Abstract— **This project aims to develop a robust model for Human Activity Recognition (HAR) using deep learning techniques. Given the significance of HAR in various applications such as surveillance, healthcare, and human-computer interaction, our study employs convolutional neural networks (CNNs) and transfer learning models like MobileNetV2 and EfficientNetB7 to classify activities in images. Results demonstrate the potential of EfficientNetB7 in achieving higher accuracy, offering insights into model performance and future research directions.**

I. Introduction

Human Activity Recognition (HAR) stands at the intersection of computer vision and human-computer interaction, serving an important role in numerous applications ranging from healthcare monitoring to intelligent surveillance systems. By enabling machines to understand and interpret human actions from visual data, HAR facilitates several interactions between humans and technology, augmenting the capabilities of automated systems to respond to human needs and behaviors effectively.

In this project, we worked on the recognition of Human activities through the use of deep learning, leveraging the capabilities of convolutional neural networks (CNNs) and transfer learning to classify images into predefined categories of human activities. We started with a baseline simple CNN model and progressively incorporate more sophisticated models, including ResNet50, MobileNetV2 and EfficientNetB7, through transfer learning. These models, renowned for their efficiency and efficacy in image classification tasks, are fine-tuned to the nuances of our HAR dataset, with a keen focus on optimizing performance while balancing computational constraints.

The models are further evaluated using various evaluation metrics including accuracy, loss values, precision, recall, and F1-scores to evaluate the model comprehensively across the spectrum of activity classes.

II. Background

The advent of deep learning has ushered in a new era in the field of computer vision, with Convolutional Neural Networks (CNNs) at the forefront of this revolution. CNNs, renowned for their prowess in image recognition tasks, form the backbone of our exploration into Human Activity Recognition (HAR). This section delves into the foundational aspects of CNNs and extends the discussion to encompass advanced architectures like ResNet50, MobileNetV2, and EfficientNetB0, which have been pivotal in our project.

A. Convolutional Neural Networks(CNNs)

Convolutional Neural Network work by using convolutional layersalong with filters to capture spatial hierarchies of features in images, from simple edges to complex objects. Pooling layers interspersed among convolutional layers reduce dimensionality, enhancing computational efficiency and feature robustness. Fully connected layers towards the end integrate these features for classification tasks. The simplicity yet effectiveness of CNNs make them an ideal starting point for HAR tasks, providing a baseline against which more complex models can be measured.

B. ResNet50

Residual Networks introduce the concept of skip connections to solve the problem of vanishing gradient problem in deep networks. ResNet50, a 50-layer variant of the ResNet model series, allows for the training of substantially deeper networks by using these skip connections to add the input of a layer to its output that ensure the flow of gradients during backpropagation. This architecture's ability to learn deep representations makes it a strong candidate for HAR.**[1]**

C. MobileNetV2

Designed with mobile and edge devices in mind, MobileNetV2 stands out for its balance between efficiency and accuracy. It introduces the inverted residual structure with linear bottlenecks, where the convolutional layer inputs and outputs are thin, and expansion layers in between allow for a richer feature set. This architecture's lightweight nature, without significant compromise on performance, makes it especially suited for applications where computational resources are limited.**[2]**

D. EfficienetB0

EfficientNetB0, part of the EfficientNet family, is known for its compound scaling method which uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. This approach results in models that achieve higher accuracy with fewer parameters. EfficientNetB0, being the baseline model of this family, provides an excellent trade-off between efficiency and accuracy, making it an ideal choice for HAR tasks that demand both high performance and computational efficiency.**[3]**

III. Dataset and Preprocessing

The core of any deep learning project is the dataset that is used to train it. Therefore, a good dataset is of high importance. Our project is trained on a dataset consisting of over 12,000 labeled images, spanning 15 diverse classes of everyday human activities. These classes encompass a wide range of actions, from mundane tasks such as 'eating' and 'texting' to more dynamic activities like 'running' and 'dancing'. The images are sourced from a public repository that reflect a rich tapestry of real-world scenarios, capturing the essence of human interactions and movements in varied settings and postures**[4].**

A. Dataset Characteristics

The dataset is divided into two primary directories: a training folder and a test folder, each containing a vast array of images depicting various human activities. The training folder is accompanied by a training_set.csv file, which maps each image to its corresponding activity label using it's filename. The images are in RGB format, offering a full spectrum of color information that is crucial for identifying nuanced differences in human activities. The test folder contains images intended for model evaluation and is associated with a testing_set.csv file that lists the filenames of the test images without their activity labels. This setup mirrors real-world applications where the true labels are unknown and provides an unbiased metric for assessing the model's predictive accuracy.

Fig. 1. *Some sample preprocessed images with labels*

Upon further checking the distribution of the classes in the dataset, the dataset contains equal distribution of images for each class.

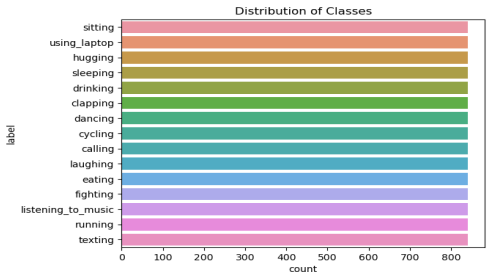

Fig. 2. *Distribution of classes in the dataset*

B. *Preprocessing Steps*

- **Resizing**: All images are resized to a uniform dimension of 224x224 pixels. This standardization is crucial for batch processing and ensures that the models have a consistent input shape.
- **Normalization**: Pixel values, originally ranging from 0 to 255, are normalized to fall within the 0 to 1 range. This normalization step aids in the convergence of the model by smoothing the optimization landscape.
- **Data Augmentation**: To reduce the risk of overfitting and enhance the model's ability to generalize to unseen data, we employ several data augmentation techniques. These include:
    - **Horizontal Flipping:** Images are flipped along the vertical axis, reflecting real-world scenarios where activities can be performed in mirrored orientations.
    - **Rotation:** Images are randomly rotated within a range of ±30 degrees, introducing variability that simulates different perspectives.
    - **Zoom:** Random zooming in and out of images within a specified range (up to 10%) helps to mimic the effect of varying distances between the subject and the camera.
    - **Shear Transformation:** A shear range of up to 20% is applied, skewing the image and simulating a change in viewpoint.

These preprocessing steps increase the dataset's diversity and introduce a level of complexity that challenges the model to learn robust and discriminative features.

- **Data Splitting**: The training dataset is partitioned into training and validation sets. The training set, comprising 80% of the data, is used to train the models. The remaining 20% is used for validation, which guides hyperparameter tuning and model selection. The testing set is reserved for the final evaluation of model performance.

C. *Data Pipeline Optimization*

Feeding the data to the model for training is streamlined by leveraging TensorFlow's "**ImageDataGenerator**" class, which facilitates on-the-fly image preprocessing and augmentation. This effectively reduces the model's training time and memory footprint. The "**flow_from_dataframe**" method is utilized to map the images to their labels present in the **training_set.csv** file, ensuring a seamless integration of the dataset's structure with the model's training requirements.

IV. Methodology

The methodology for our project included the deployment and fine-tuning of sophisticated deep learning architectures by performing hyperparameter optimization using grid search to find the best hyperparameters for the model. The chosen architectures include a custom-made CNN model and implementing different transfer learning models using ResNet50, MobileNetV2, and EfficientNetB0 models, each selected for its unique attributes in relation to image classification tasks.

Early stopping was added to each model during training, by monitoring the validation loss with patience level of 5.

A. *Custom CNN model*

Architecture: The foundation model begins with a sequential CNN architecture consisting of five convolutional layers. Each convolutional layer uses 3x3 kernels, with the depth of filters doubling at each subsequent layer, starting from 32 to 64, and then 128, to progressively extract more complex features. Following each convolutional layer is a max-pooling layer with a 2x2 pool size, that reduces spatial dimensions and computational load while preserving essential features. Manual hyperparameter tuning was done to find the best hyperparameters for the model.

Hyperparameter	Best Value
Dense Layer neurons	256
Activation function for convolutional layers and Dense layers	relu
Activation function for final layer	softmax
Optimizer	Adam
Learning rate	0.0001

Softmax function at the last layer was used for it's use case in multi-class classification as it outputs the probability of the image belonging to each of the class.

*Model Architecture**[5]***

Layer (type)	Output shape	Parameters
Conv_2D_14	(None, 222, 222, 32)	896
max_pooling2d_14 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_15 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_15 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_16 (Conv2D)	(None, 52, 52, 64)	36,928
max_pooling2d_16 (MaxPooling2D)	(None, 26, 26, 64)	0
conv2d_17 (Conv2D)	(None, 24, 24, 128)	73,856
max_pooling2d_17 (MaxPooling2D)	(None, 12, 12, 128)	0
conv2d_18 (Conv2D)	(None, 10, 10, 128)	147,584
max_pooling2d_18 (MaxPooling2D)	(None, 5, 5, 128)	0
flatten_9 (Flatten)	(None, 3200)	0
dense_43 (Dense)	(None, 256)	819,456
dense_44 (Dense)	(None, 128)	32,896
dense_45 (Dense)	(None, 15)	1,935

Training: The model was trained from scratch, with random weight initialization. The training process leverages data augmentation techniques applied during preprocessing to enhance the model's ability to generalize and prevent overfitting.

B. *ResNet50*

Architecture: We utilize a pre-trained ResNet50 model, excluding the top layer, and append custom dense layers to adapt the model to our classification task. The custom layers include a global average pooling layer, a dense layer consisting of 256 full-connected neurons and a final dense layer consisting of 25 fully-connected neurons, that output the class of the input image.

*Model Architecture**[5]***

Layer (type)	Output shape	Parameters
ResNet-50 Base	(None, 7, 7, 2048)	Pre-trained
GlobalAveragePooling2D	(None, 2048)	0
Dense_1 (Dense)	(None, 256)	524,544
Dense_2 (Dense)	(None, 15)	3,855

Fine-tuning: Initially, all layers in the pre-trained model are frozen, and only the custom layers are trained. Subsequently, the last 25 residual blocks are unfrozen, and the entire model is fine-tuned with a reduced learning rate to refine the high-level features for the HAR task.

Hyperparameter tuning: The number of fully-connected neurons in the dense_1 layer was decided by using grid search to find the best value among **32,64,128 and 256 neurons**.

Optimization: Fine-tuning employs the Adam optimizer with a lower learning rate (e.g., 1e-5) to prevent drastic updates that could destabilize the learned weights. The loss function used is categorical **cross-entropy**. These parameters were set after employing hyperparameter optimizations on learning rate (with different values tested), activation functions (relu, sigmoid, softmax etc), loss function and optimizer to be used.

C. *MobileNetV2*

Architecture: The customized layers added to this model vary from ResNet-50 in the way that the Dense layer is made up of 1024 full connected neurons instead of 256, in case of ResNet-50.

*Model Architecture**[5]***

Layer (type)	Output shape	Parameters
MobilenetV2 Base	(None, 7, 7, 1280)	Pre-trained
GlobalAveragePooling2D	(None, 1280)	0
Dense_1 (Dense)	(None, 1024)	13,11,744
Dense_2 (Dense)	(None, 15)	15,375

Fine-tuning: Similar to ResNet50, the MobileNetV2 model pre-trained on ImageNet is adapted by excluding the top layer. The model is initially trained with frozen weights, followed by unfreezing of last 20 layers for fine-tuning, ensuring a careful adjustment of feature representations.

D. *EfficientNetB0*

Architecture: Similar custom-layers as the MobileNetV2 model are added to this model with global average pooling layer, 1024 neuron fully connected neuron layer(with relu activation) before the final classification dense layer with softmax activation function.

*Model Architecture**[5]***

Layer (type)	Output shape	Parameters
EfficientNetB0 Base	(None, 7, 7, 1280)	Pre-trained
GlobalAveragePooling2D	(None, 1280)	0
Dense_1 (Dense)	(None, 1024)	13,11,744
Dense_2 (Dense)	(None, 15)	15,375

Fine-tuning: The model undergoes a phased training approach, where initial training with frozen base weights is followed by selective unfreezing of the last 20 layers and fine-tuning of the entire network to adapt it to the dataset.

V. Results

Our exploration into different deep learning models for the human activity recognition let us to discover insightful results and increased our understanding about these deep learning models. Let us discuss these results-:

A. *CNN*

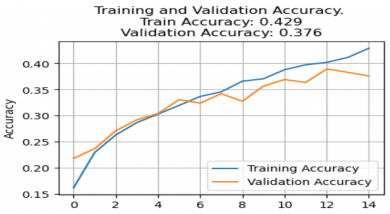

Fig. 3. *Accuracy vs epoch graph*

The custom CNN model created by us provided a modest foundation in accuracy. It model performed with an accuracy of 42.9% on the training dataset, while it gave around 37.6% accuracy on the validation dataset. The validation accuracy peaked at around 38%.

Fig. 4. *Loss vs epoch graph*

Observing the Training loss and validation loss vs epoch, we can observe that validation loss started to diverge from training loss, indicating signs of overfitting. The performance of the model highlights the need for more complex architectures.

B. *ResNet50*

The implementation of ResNet50 improved the performance, leveraging deep residual learning to enhance feature extraction and classification capabilities.

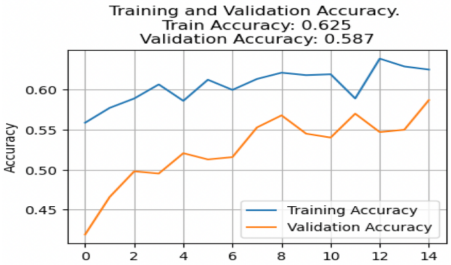

Fig. 5. *Accuracy vs epoch graph*

The model reached training accuracy of 62.5% and 58.7% accuracy on the validation dataset. Observing the loss vs epoch graph, the training loss continues to decrease. The validation loss starts to fluctuate near the epoch 10, although decreasing. Therefore, it was stopped by the early stopping criteria. The absence of peaks in the validation loss curve rules out the likelihood of large, destabilizing learning rate effects.

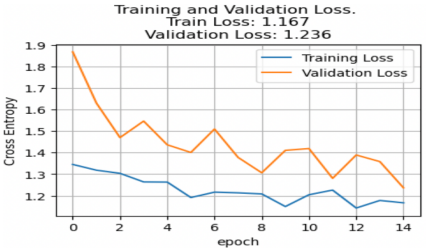

Fig. 6. *Loss vs epoch graph*

It suggests the need for exploring more architectures for even better accuracy.

C. *MobileNetV2,*

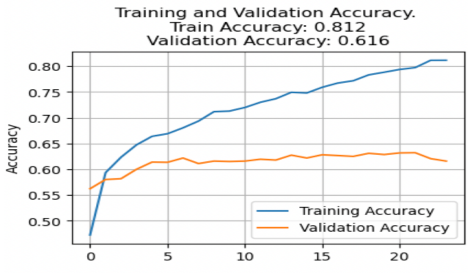

Fig. 7. *Accuracy vs epoch graph*

The accuracy vs epoch curve for the MobileNetV2 model shows a steep increase in training accuracy from starting, indicating that model is rapidly learning from the training data, indicating the model's weight parameters adjusting well, thus having a good learning rate. The validation accuracy increases in the beginning, however starts to stabilize at around 62%. It indicates the start of overfitting. Thus, the model is stopped at epoch 24 and returned to epoch 19 using early stopping.

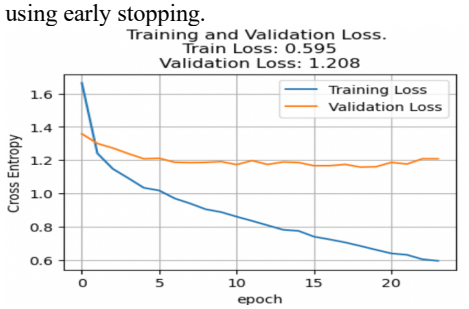

Fig. 8. *Loss vs epoch graph*

The loss curve shows similar trend. The training loss keeps decreasing while the validation loss stops decreasing after some epochs. However, the model performs better than other models on the validation dataset.

D. *EfficientNetB0,*

The EfficientNetB0 model, through its compound scaling methodology, achieved superior accuracy, compared to other models, effectively capturing complex patterns in the data.

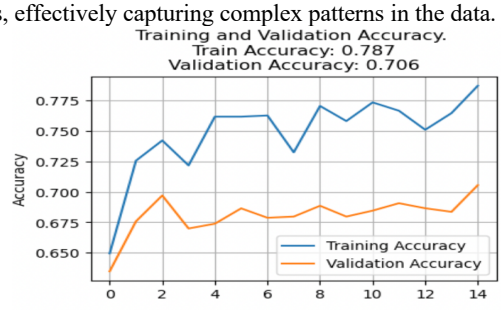

Fig. 9. *Accuracy vs epoch graph*

The training accuracy improves significantly and consistently, reaching approximately 78.7%, suggesting that the model is effectively learning from the training data. The validation accuracy shows more variability but also a positive trend, stabilizing around 70.6%. This smaller gap compared to earlier models indicates better generalization to unseen data.

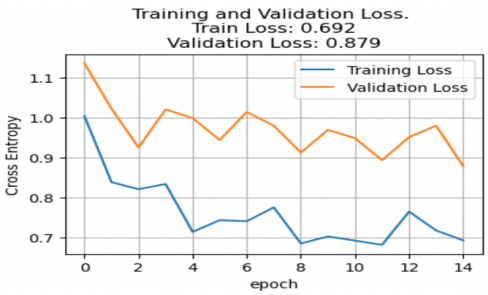

Fig. 10. *Loss vs epoch graph*

The loss graphs corroborate this, with training loss decreasing to about 0.692 and validation loss showing fluctuations before settling near 0.879. The convergence of these metrics indicates a more balanced fit than previous models.

E. *Ensemble Learning (ResNet50 + Mobilenet V2 + EfficientNetB0)*

After training all transfer models, we combined all the three transfer learning models to create the ensemble learning model. After compiling the model, we predicted the output of the model on the test dataset. The model showed good results.

Fig. 11. *Ensemble model predictions on test dataset*

The predictions on the test dataset show that the ensemble learning model is better at generalizing than the individual transfer learning models.

VI. Conclusion

In this project, we explored the architectures of various deep learning models to find their suitability for the purpose of Human Activity Recognition (HAR). Through the iterative process of building, evaluating, and refining models, we understood and analyzed the capabilities and limitations of each architecture.

Our journey from a simple CNN to the sophisticated EfficientNetB0, supplemented by an innovative ensemble approach, illustrates the dynamic interplay between model complexity, computational efficiency, and predictive performance. The superior results of the EfficientNetB0 and the ensemble model underscore the value of scalable architectures and diversity in model predictions, respectively.

The success of the ensemble model on the test dataset illuminates the path forward for deploying deep learning-based HAR systems in real-world applications, from surveillance to assistive technologies.

References

[1] https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f

[2] https://paperswithcode.com/method/mobilenetv2

[3] https://medium.com/image-processing-with-python/efficientnetb0-architecture-stem-layer-496c7911a62d

[4] https://www.kaggle.com/datasets/meetnagadia/human-action-recognition-har-dataset

[5] ECE9039 Project Python Notebook

Appendix

Link to the github repository -
https://github.com/gsing352/ECE9039-Human-Activity-Recognition/tree/main

Dataset Link -
https://www.kaggle.com/datasets/meetnagadia/human-action-recognition-har-dataset

Contribution Statement -
In the successful completion of the project, each group member played a crucial role. This project is a collaborative effort, with each member contributing as follows-:

Group Member	Student Number	Project Contribution	Project Report Contribution
Gagan Deep Singh	251368991	Data Preprocessing, ResNet50	Introduction, Dataset and Preprocessing, Results(ResNet50)
Gurjot Singh	251362186	Data Preprocessing, Mobilenet V2	Background(CNN, ResNet50), Methodology(CNN, Resnet50), Results(MobilenetV2)
Mansimar Singh Bhatia	251397046	Data Preprocessing, CNN	Background(MobilenetV2, EfficientNet-B0), Methodology (MobilenetV2, Efficientnet-B0), Results(CNN)
Sherry Ahuja	251383252	EfficientNetB0, Ensemble Learning	Dataset and Preprocessing, Results(Efficientnet-B0, Ensemble model), Conclusion