CS747 - Project Final Report

Swabhi Papneja [1], Sai Raswanth U M [2], Ramanuja Phani Vishnu Teja Kandalam [3] *

Project Type:

Reproducibility challenge

Selected Paper:

Why Can't I Dance in the Mall? Learning to Mitigate Scene Bias in Action Recognition.

Authors:

Jinwoo Choi*, Chen Gao*, Joseph C. E. Messou*, Jia-Bin Huang* (* Virginia Tech)

While Convolutional Neural Networks (CNNs) excel in recognizing and extracting features from images, they often struggle with transfer learning tasks. This is because they frequently focus on learning unintended features, such as the scene background, rather than the actions themselves. The paper proposes a novel architecture that penalizes the model for learning scene-related features in videos during pre-training, effectively reducing scene bias.

We reproduced the paper's approach by implementing the proposed loss functions and optimizers and tested its transfer learning capability across three different action recognition datasets. Our findings demonstrate that the debiasing loss function enables models to generalize better, outperforming a baseline model in transfer learning tasks.

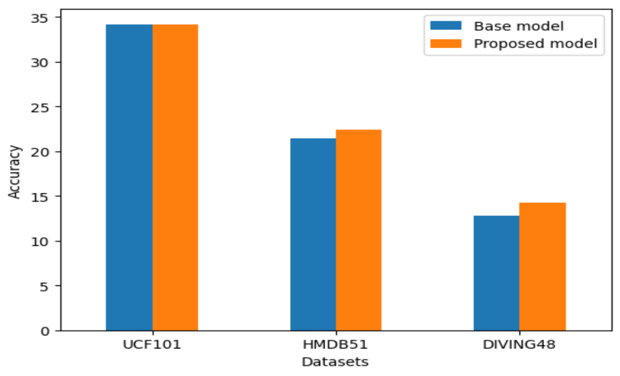


Figure 1: Accuracy of proposed method on Transfer learning tasks

*

1: G01441441, spapneja,
2: G01447320, supalami,
3: G01447547, rkandala

1 Introduction

Convolutional Neural Networks (CNNs) have been and continue to be transformative in the field of deep learning due to their ability to handle large-dimensional data and their efficient feature extraction techniques. Although they perform well with standard deep learning methods for training and testing models, it is evident that they also tend to learn unintended features from the input images. This results in an inherent bias in the model as it learns features that are not relevant or important. This issue becomes particularly problematic when the trained model is used for transfer learning tasks. Furthermore, if the base dataset used for pre-training the model contains specific biased features, this bias can propagate. For example, if we aim to build a model for vehicle detection and the base dataset contains cars only in red, it would create challenges for transfer learning tasks that rely on these pre-trained weights.

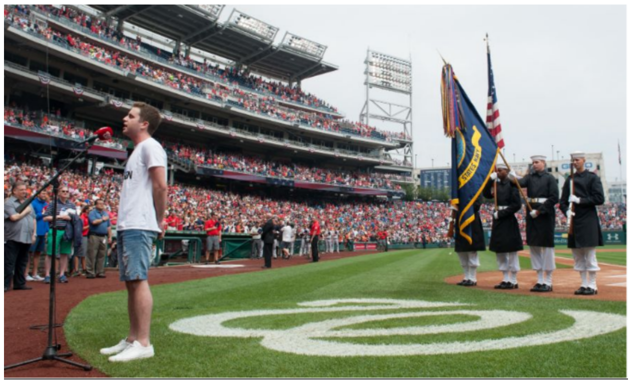


Figure 2: Example Scene

The authors of [1] focused on addressing the challenge of scene bias in the Action Classification task, where the goal is to predict the action performed in a given video. For instance, in Fig. 2, much of the image is dominated by scene-related features, such as the sky, playground, or crowd. This strong presence of the scene (e.g., a playground) can cause the model to predict the action as ”baseball” or another sport based solely on the scene, rather than the actual action being performed. To overcome this, the pre-training architecture must be designed to prioritize extracting action-specific features over scene-related features.

However, extracting features that are truly representative of an action is challenging because many actions are inherently correlated with the scene in the video, which can influence the features the model learns. To tackle this, the authors proposed a novel pre-training network architecture designed to extract action-relevant features while penalizing the model for focusing on unintended scene features. Alongside the commonly used action classification loss, they introduced two new loss functions: **(i) Scene Adversarial Loss and (ii) Human Mask Confusion Loss.**

These loss functions guide the model to focus solely on action-related features. Specifically, the model is rewarded for correctly predicting actions from videos and penalized when the extracted features rely on scene-related information or are insufficiently dependent on the action. This training strategy produces a model with pre-trained weights that are optimized to represent actions, improving the model's effectiveness in downstream transfer learning tasks. This document is organized as follows:

....

2 Details of the Approach

This section delves into the pre-training network architecture, introduces the proposed loss functions, outlines the optimization approach employed during pre-training, and concludes by discussing the transfer learning application of the proposed model.

2.1 Methodology/Architecture

The approach proposed for pre-training the feature extractor to learn the actions that persist in the given images (video) is presented in Figure 4. In the training phase, the feature extractor should be trained with human-masked frames in addition to the regular frame data which helps it to quantify the human action feature representation in the model prediction. The green arrows in Fig 4 represents the calculation of plain cross-entropy loss that can help the feature extractor in prediction of better action labels. In addition to the action classification loss, the author proposed two new loss functions to mitigate the bias. While the green path represents the action classification loss, the red and blue paths represent the scene and human mask confusion losses respectively. The model employs a special layer called **Gradient reversal** [2] during back-propagation of L_scene to identifies the weights that represent scene data within feature extractor and multiplies their gradients by a negative scalar. This discourages the feature extractor to learn features that are representative of the scene in a given frame. After scene classification, the feature extractor and the action classifier are reused with human-masked frames to calculate L_hmask. This loss helps in measuring the confidence of the model without action data. Each of these loss functions and their formulas are explained below:

2.2 Loss Functions

2.2.1 Action Classification loss:

This is a binary cross-entropy loss that was used to update the weights of the feature extractor based on the actions predicted from the input frames. It is calculated using the following equation:

$$L_{\text{action}} = -\mathbb{E}_{(x,y)\sim(X,Y)} \sum_{k=1}^{N} y_k \log f_{\theta_A}(G_{\theta_f}(x))$$

where X represents the input frame, G_{θ_f} represents the feature extractor, N is the number of action classes and $y \in Y$ where Y represents the target action labels of the dataset.

2.2.2 Scene classification adversarial loss:

This loss function makes sure that the features that were extracted from the frames are not entirely dependent on the scene data in the images. The loss is calculated using the following equation:

$$L_{\text{scene}} = -\mathbb{E}_{(x,p)\sim(X,P)} \sum_{m=1}^{M} p_m \log f_{\theta_S}(G_{\theta_f}(x))$$

where X represents the input frame, G_{θ_f} represents the feature extractor, f_{θ_s} represents the scene classifier, M is the number of scene types and $p \in P$ where P represents the target scene labels of the dataset. This is an adversarial loss as the features in action classifier (θ_f) increases the loss, while the parameters of scene classifier (θ_s) tried to decrease the loss.

2.2.3 Human mask Confusion loss:

This loss helps the architecture to understand the role of the human subjects on the predictions. After making a prediction without human masks, we use the same feature extraactor and action classifier to generate prediction with the human masked frame. The confidence in the prediction of the model when the human subjects are masked out resembles the impact of human subjects in the input frames. This is measured by the entropy of the prediction and the entropy of the model prediction should be high, representing the lack of confidence in prediction of a class. The loss is calculated using the following equation:

$$L_{\text{hmask}} = -\mathbb{E}_{x_{\text{hm}} \sim X_{\text{hm}}} \sum_{k=1}^{N} f_{\theta_A}(G_{\theta_f}(x_{\text{hm}})) \log f_{\theta_A}(G_{\theta_f}(x_{\text{hm}}))$$

where X_hm represents the human-masked input frame, G_{θ_f} represents the feature extractor, f_{θ_A} represents the action classifier, and N is the number of action classes.

2.3 Optimization

We can use the above loss functions and define the optimization as follows, when the input video is x (with human actions) the optimization is:

$$L(\theta_f, \theta_S, \theta_A) = L_{\text{action}}(\theta_f, \theta_A) - \lambda L_{\text{scene}}(\theta_f, \theta_S)$$

$$(\theta_f^*, \theta_A^*) = \arg\min_{\theta_f, \theta_A} L(\theta_f, \theta_S^*, \theta_A)$$

$$\theta_S^* = \arg\max_{\theta_S} L(\theta_f^*, \theta_S, \theta_A^*)$$

Here λ is a hyperparamter that can determine the impact of the scene classifier.

When the human subjects in the video are masked out, we have to maximize the human mask confusion loss as the entropy of the model prediction should be maximum in an ideal case.

$$(\theta_f^*, \theta_A^*) = \arg\max_{\theta_f, \theta_A} L_{\text{hamsk}}(\theta_f, \theta_A)$$

2.4 Transfer Learning

After pre-training the model using the above loss functions, it equips the feature extractor with weights that can extract action features of a frame rather than the scene data. In the proposed methodology, the pre-trained feature extractor can be further used for other action classification tasks such as temporal action detection and spatio-action identification (Figure 3).

3 Experimental Results:

3.1 Datasets

We are working with four video datasets related to action understanding tasks for this reproducibility study. The research proposed a debiasing that will be pretrained on MiniKinetics dataset and then evaluated on HMDV51, Diving48 and UCF101 datasets. We started with analysing these datasets. The number of videos in every dataset is listed in the adjoining Table 1. We have explained details about these datasets below:

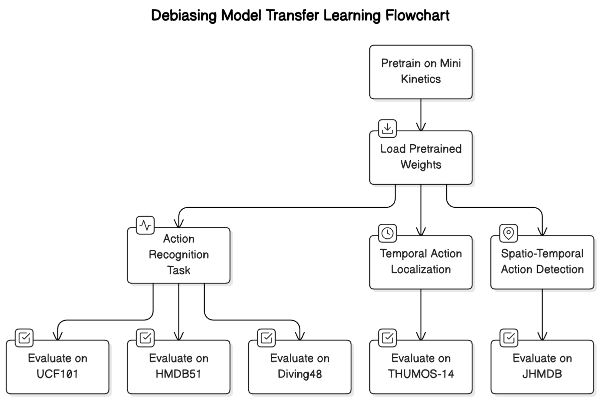


Figure 3: Transfer learning using the feature extractor

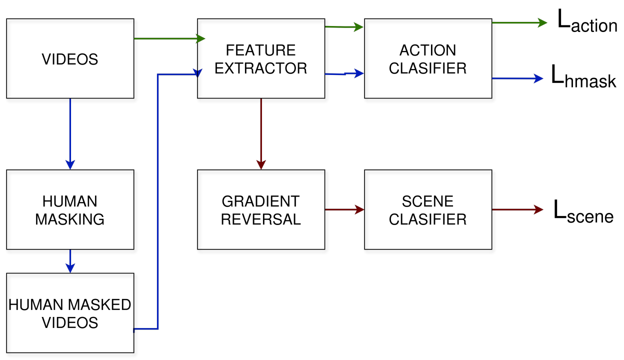


Figure 4: Pre-training model architecture

Dataset	**# of _samples**	**# of _classes**
Mini Kinetics 200	80,000	200
HMDB51	6,766	51
UCF101	13,320	101
Diving48	18,000	48

Table 1: Summary of Datasets with Sample Counts and Number of Classes

3.1.1 Mini Kinetics 200:

The Mini Kinetics dataset is a smaller, widely used subset of the larger Kinetics dataset for action recognition. The dataset comprises 200 distinct activity classes, including actions such as dancing, surfing, and arm wrestling, among others. This dataset is derived from a compilation of YouTube videos publicly available on the internet. Key features of this dataset are as follows:

1. 200 Action Classes, with balanced categories, covering a broad range of human activities.
2. 400 Clips per Class, totalling around 80,000 video clips.
3. The video clips are typically 256 x 256 pixels with a variable frame rate, preprocessed for uniformity across clips.
4. Each video clip lasts for 10 seconds on average, capturing enough context for action recognition.
5. **Challenges:** The dataset has high variability in backgrounds, lighting, and viewpoints, presenting challenges in distinguishing similar actions.

However, due to the dynamic and evolving nature of YouTube's content, many videos have been removed, set to private, or otherwise become inaccessible for download, presenting significant challenges in developing our base model. For pre-training purposes, we initially limited our model to the **100 classes** with the highest video availability. Even in that 100 classes, each class has an average of only 275 videos available to download. Nonetheless, in the final analysis, we aim to train the Mini Kinetics model using the complete dataset to enhance its generalization and robustness.

3.1.2 [6]HMDB51:

The [6]HMDB51 dataset is another popular benchmark for action recognition and detection in videos, with 51 Action Classes across categories such as General Facial Actions, General Body Movements, and Interactive Actions. There are 6,766 Video Clips with varying resolutions, and diverse sources, including movies and YouTube. There are roughly 3,570 training and 1,530 testing videos per split. [6]HMDB51's variety in action types, video quality, and background conditions provides a robust test for action recognition models.

3.1.3 UCF101:

The UCF101 dataset is widely used for action recognition and detection in videos, featuring 101 Action Classes across categories like Human-Object Interaction, Body-Motion Only, and Sports. There are a total of 13,320 Video Clips with resolutions of 320 x 240 pixels and a frame rate of 25 FPS, each lasting 3 to 10 seconds. Each split has 9,537 training and 3,783 testing videos. There are a few challenges working with this dataset as well, including high background variability, camera movements, and inter/intra-class similarity making it difficult for models to distinguish actions accurately.

3.1.4 Diving48:

The Diving48 dataset consists of videos showcasing 48 distinct diving positions, selected for its focus on capturing the primary actions within each video rather than the background. This characteristic emphasizes the importance of the movement itself, allowing models to better generalize across varied contexts and environments. The dataset is readily accessible and organized into well-defined classes for training and validation, making it particularly suitable for action recognition tasks. By focusing predominantly on the action of diving, this dataset provides valuable insights into dynamic human movements, which are beneficial for refining the accuracy of the model in action-specific recognition tasks.

3.2 Data Preprocessing:

Dataset	Size	Frames Extracted
Mini-Kinetics 200	143 GB	10,485,759
HMDB51	2.1 GB	632,599
UCF101	6.8 GB	2,483,295
Diving48	9.7 GB	2,465,263 + 216,808

Table 2: Dataset sizes and frames extracted for each dataset used in the study.

3.2.1 Downloading and Preparing Data:

Table 2 shows that Dataset sizes and frames extracted for each dataset used in the study. The detailed explaination of the Data Downloading and Preparing process is given below:

1. **Mini Kinetics 200:** [5]

   - The Mini Kinetics 200 (MK200) dataset, which consists of 80,000 video IDs across 200 classes, was used as a primary dataset.
   - The source for this dataset is YouTube. To download the videos efficiently, we leveraged the yt-dlp Python module. We divided the 80,000 video IDs into groups corresponding to the 200 classes and implemented parallel video downloading scripts.
   - This approach utilized the multi-core processing capabilities of the provisioned system to ensure rapid and efficient data acquisition. After downloading the videos, frames were extracted at a rate of one frame per second using OpenCV (cv2) and ThreadPoolExecutor for parallelism.
   - These frames were organized into respective class directories, with each directory named numerically from 0 to 199, representing the 200 classes in the MiniKinetics 200 dataset.
   - Figure 5 shows the distribution of samples across classes in the Mini-Kinetics dataset. Darker cells indicate classes with more samples, while lighter cells highlight underrepresented classes. The visualization reveals an imbalance in the dataset, with some classes having significantly fewer samples, which may affect model training and necessitate strategies like data augmentation or class weighting.

2. **UCF101 [8] , Diving48 [7], and HMDB [6] Datasets**: The UCF101, Diving48, and HMDB datasets were obtained from their respective official sources. Once downloaded, the datasets were uploaded to the Hopper user memory. A Python script was developed to extract

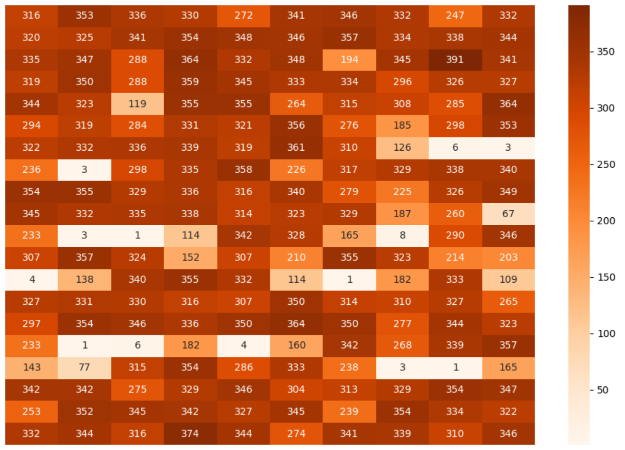


Figure 5: Number of samples per class in Minikinetics dataset

every frame from the videos, capturing all frames as they naturally occur in the video (based on the original frame rate). The extracted frames were then organized into class-specific directories. For frame extraction and preprocessing, the mmcv library was utilized, enabling dynamic resizing to accommodate varied input/output requirements. These processes were implemented with reference to the guidelines provided in the MMAction2 toolkit.

3.2.2 Frame Extraction:

After gathering all the video content in all the datasets, each video has to be converted into frames so that it can be given as input to the CNN. For this purpose, we used the scripts from [3] to extract frame data from each video. A single frame is chosen per second from the video to match the time and computation constraints posed by these large datasets. After converting each video to frames, 16 frames are randomly sampled from each video to maintain the consistency in the input dimensions of each sample.

3.2.3 Masking Mini Kinetics Data for Debiasing

The Mini Kinetics dataset was used to pretrain the model. Subsequently, the model was fine-tuned for debiasing, requiring masked data based on the Mini Kinetics pretraining. Masking was performed using a [4]ResNet-50 model pretrained on the ImageNet dataset. Due to the computational expense of this task, a subset of 16 frames per video ID was selected for masking, rather than processing all extracted frames.
Challenges: Masking the frames required significant computational resources. Using a GPU with 10 GB of memory on Hopper, the estimated time to mask all frames was approximately 400 hours. To mitigate this overhead, the number of frames considered for masking was strategically reduced, to balance computational feasibility and data quality requirements.

3.2.4 Generating Pseudo Labels:

We generated the pseudo scene labels for our scene classification component in the debiasing model. We created a json file in which every video was associated with a predicted scene label.

This is done by using the Places365 model, a pre-trained [4]ResNet18 specifically designed for scene recognition. For each video, we sample a few representative frames, preprocess them, and pass them through the Places365 model to obtain probabilities for different scene categories. The average probabilities across the selected frames are used to determine the most likely scene label. Table 3 shows the pretrained model architecture used for this task.

3.3 Implementation Details

3.3.1 Baseline Model

We implemented a baseline model to train our feature extractor using the [4]ResNet-3D (R3D-18) architecture to extract spatiotemporal features from the video frames for action recognition. First, we have trained this model on Mini Kinetics 200 dataset then we have evaluated our pre-trained model on three datasets - [6]HMDB51, UCF101 and Diving48. The key components of the Baseline Model are explained below:

1. **Feature Extractor**: We kept the main architecture of the [4]ResNet-3D model the same for the feature extractor with the final fully connected (FC) layer removed. We also added an adaptive average pooling layer that takes the features extracted from the video frames and reduces them into a fixed size, making them ready for the final classification step without altering the main structure of the model.

2. **Action Classifier:** We added a separate linear classifier head to map the extracted features to action categories. For Mini Kinetics, this classifier outputs logits corresponding to 200 classes. For evaluation datasets UCF-101, [6]HMDB-51, Diving48, the classifier outputs logits for 101, 51, and 48 action classes respectively.

3. **Custom Dataset Preparation:** We created a custom dataset, FrameDataset to load video frames from specified directories and applied preprocessing transformations. Data was split into training, validation, and testing sets using a stratified sampling approach to ensure balanced class representation. We maintained the data structure to be uniform accross all datasets, where we have directories with class labels, that included all the videos directories belonging to that class, and the video directories included all the frames extracted for that specific video.

4. **Frame Sampling:** In our implementation, the dimension D in the video tensors, representing the temporal depth (number of frames), is fixed to 16 by carefully selecting or padding frames from each video. We are loading all the frames using the FrameDataset class for all the videos. If a video has more frames than the chosen duration, we select the first 16 frames. This ensures a consistent number of frames for each video while capturing the start of the action. However, if the video has fewer than 16 frames, we duplicate the last frame to make up the difference. This way, all video tensors have exactly 16 frames.

5. **Preprocessing and Normalization**: The selected or the padded frames are then transformed using the following preprocessing steps: (e.g., resizing, normalization) and then stacked along the temporal dimension to form a $3 \times D \times H \times W$ tensor, where D=16.

   - We resized the frames to a consistent size of 112×112, thus representing the H and W dimensions for the video tensor.
   - We then applied a center crop of 112×112 to focus on the main subject.

- We also normalized the pixel values using the mean and standard deviation values specific to the Mini Kinetics dataset.

This approach ensures uniformity across the dataset, making it suitable for training our baselne model.

6. **Training:** We trained our baseline model on the Mini Kinetics dataset using extracted video frames. We then saved this pretrained model, and loaded it's pretrained weights to evaluate it on UCF-101, [6]HMDB-51, and Diving48 datasets. The hyperparameters used while training this model are mentioned in the Table 4: Parameter settings for the model. While training, the Features were extracted using the feature extractor and then passed to the classifier head for prediction of the action labels. The loss was computed using cross entropy loss, propagated backward, and the model's parameters were updated using the SGD optimizer. After every epoch, we recorded training accuracy and loss metrics at the end of each epoch.

Usage	**Model**
Base model architecture	ResNet18
Human masking model	ResNet50 *
Pseudo Scene label generator	ResNet18 *

Table 3: Model usage details.

3.3.2 Debiasing Model

The objective of this debiasing model is to pretrain the feature extractor to reduce the invariant features related to the scene. This model addresses the issue of scene bias by multi-task learning of three different tasks in every epoch: action classification for original videos, scene classification for original videos, and action classification for human masked videos. By combining these components, the model learns to prioritize meaningful action features while minimizing it's bias towards the background information. The key components of this model are explained below:

1. **Feature Extractor:** Similar to the baseline model, our feature extractor for the debiasing model is also built upon the 3D [4]ResNet-18 architecture. It processes input video tensors to extract spatiotemporal features by applying convolutional layers, batch normalization, and pooling operations. The complete architecture of our feature extractor is explained below:

    - Input: Video tensor of shape B x 3 x 16 x 112 x 112 (Batch size, RGB channels, Frames, Height, Width), where B is our batch size.
    - Conv3D Layer (7x7, stride=(1, 2, 2), padding=3): Outputs B x 64 x 16 x 112 x 112.
    - BatchNorm3D + ReLU Layer: Normalizes and applies activation.
    - MaxPool3D Layer (3x3x3, stride=2, padding=1): Reduces dimensions to B x 64 x 8 x 56 x 56.
    - Residual Blocks: We have used 4 residual blocks, with downsampling and the last layer downsamples the video input to (B x 512 x 1 x 7 x 7).
    - Adaptive Average Pooling: Reduces dimensions to B x 512 x 1 x 1 x 1.

- Flattening: Converts tensor to B x 512 for classification tasks, which is the output dimension of the features extracted from the feature extractor.

2. **Action Classifier:** Similar to baseline model, the action classifier takes the output of the feature extractor and predicts the action class for a video. In this component, a fully connected layer is added on top of the feature extractor to map the 512-dimensional feature vector to the desired number of action classes. It outputs the action logits with shape B x num_classes (e.g., B x 200 for 200 action classes in MiniKinetics 200 dataset). We use the same action classifier for action classification on original videos and human masked videos.

3. **Gradient Reversal:** This layer multiplies the gradient by a negative scalar (alpha), 0.5 in this case.

4. **Scene Classifier:** After the gradient reversal layer, the reversed features are passed into this multilayer perceptron (MLP) component used for scene classification from the features extracted. We maximize the scene classification loss by using the gradient reversal layer to reverse gradients during backpropagation, forcing the feature extractor to learn scene-invariant features.

Parameter	**Value**
Learning Rate	0.001
Momentum	0.9
Batch Size	32
Number of Action Classes	200
Number of Scene Classes	365
Alpha (Scene Adversarial)	0.5
Alpha (Human Mask)	0.5
Optimizer	SGD
Pre-training Epochs	10
Transfer Learning Epochs	30

Table 4: Parameter settings for the model.

5. **Custom Dataset Preparation:** In addition to the custom dataset preparation done in baseline model, we have added the support to handle masked videos in the form of image frames along with the original videos. We are also extracting the pseudo scene labels and stacking them into the final video tensor. Since we do not have the action labels for the test dataset, we have also handled the video tensor preparation in testing mode, where we will not be adding action labels to the final video tensor. We have used one-hot encoding for both action and scene labels, converting them into fixed-length vectors. Thus, we prepare data for a multi-task learning setup in debiasing model, with action classification and adversarial scene classification.

6. **Frame Sampling:** Similar to the frame sampling in the Baseline model, we are sampling 16 frames for every video for the debiasing model as well.

7. **Training and Optimzation:** Losses from all the three tasks are combined and backpropagated to update the model parameters. Optimizers like SGD with momentum and weight decay are employed to stabilize the training process.

8. **Saving Pretrained Weights:** After training the model on the Mini-Kinetics dataset, the learned weights of the feature extractor are saved as pretrained weights. These weights serve as an initialization for further evaluation on other datasets such as UCF-101, Diving48, and [6]HMDB-51.

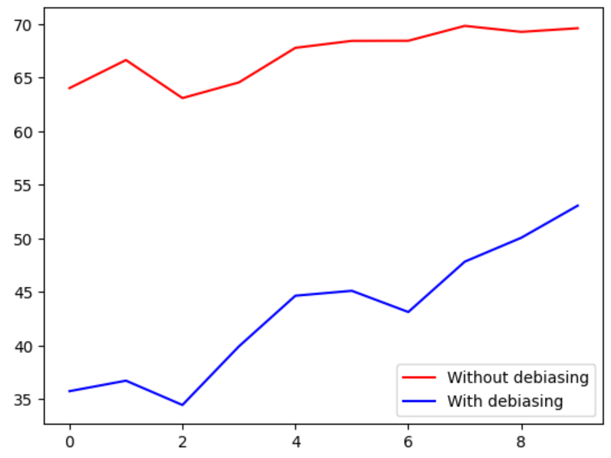
3.3.3 Transfer Learning

Next, we load the saved models from both the baseline model and debiased model pretrained on MiniKinetics dataset to initialize the feature extractor with the pretrained weights of our debiasing model before moving further with the evaluation on new datasets. For transfer learning on the evaluation datasets, we replace the final fully connected (FC) layer of the pretrained model with a new FC layer according to the number of classes in the target dataset, 51 classes for [6]HMDB51, 48 for Diving48, and 101 for UCF101 datasets. Then, we fine-tune the model on the evaluation datasets by updating its parameters through backpropagation, using cross-entropy loss. We then compare the results of the baseline model and the debiasing model on these datasets.

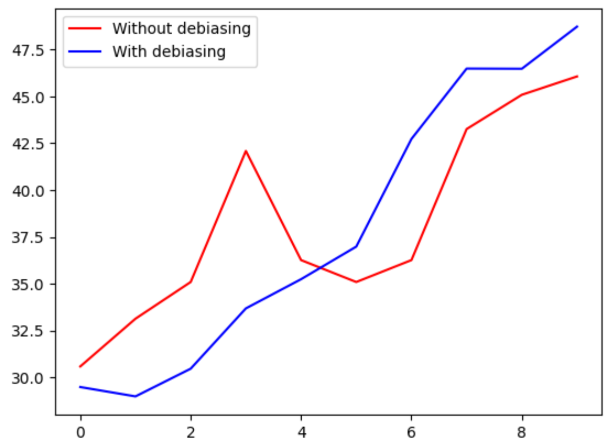
4 Results

We have pre-trained a naive base [4]ResNet18 (base model) and a model with debiasing loss for 10 epochs. Thee Train and Validation accuracy over the 10 epochs are presented in 6. The train and validation loss is presented in Fig. 7. It can be seen that the loss of model with debiasing is higher in the initial epochs due to its reluctance to learn the major scene features for prediction. But this proved to be successful in final prediction on the minikinetics validation set.

After pre-training both the models, we tested the generalization ability of both the models on three action recognition tasks. This test helped in identifying the impact of using debiasing method rather than pre-training with a simple loss function. The training and validation accuracy achieved on these three datasets over 30 epochs is presented in Figure 8.



(a) Training accuracy on Minikinetics dataset

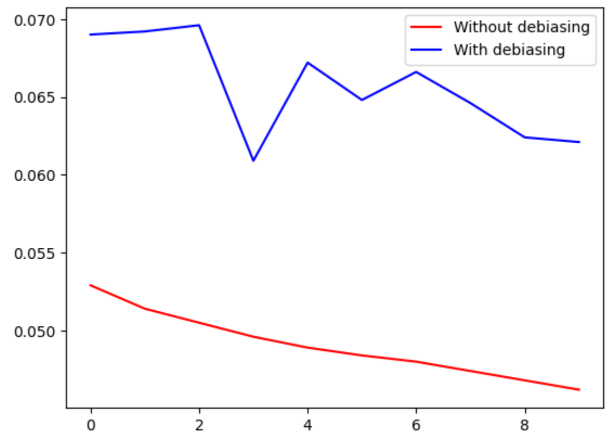


(b) Validation accuracy on Minikinetics dataset

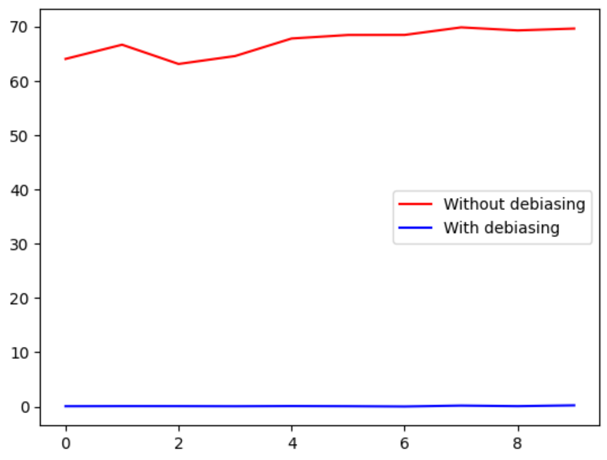
Figure 6: Training and validation accuracy on Minikinetics dataset.

5 Discussions & Conclusion

The results of the methodology demonstrates that the pre-trained model with debiasing values has actually helped in restricting the model from learning unintended features, further increaseing



(a) Minikinetics Training loss



(b) Minikinetics Validation loss

Figure 7: Training and validation loss for the Minikinetics dataset.

the ability of the model to generalize over new datasets. These results on the initial transfer learning tasks can also be applied to other problems such as spatio action identificaiton and temporal action detection, or eve both. Table 5 presents our final results of the model with and without using debiasing approach for pre-training the model. By increase in the accuracy of three different transfer learning daatasets, it is evident that the model is able to learn more action level features than scenes.

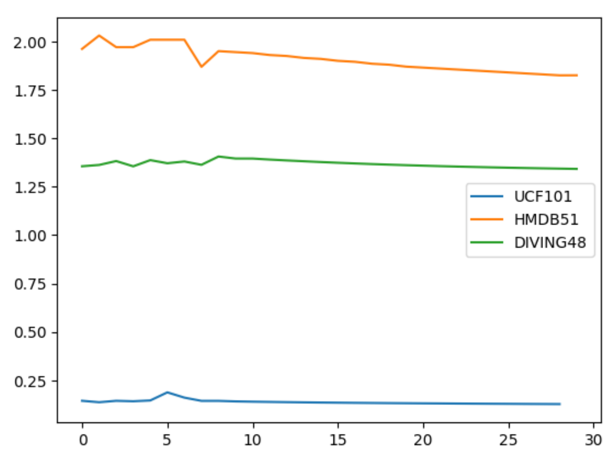
Model Approach	**Mini-Kinetics**	**UCF101**	**HMDB51**	**Diving48**
Baseline with Pretrained Weights	–	72.4	32.4	–
Baseline Pretrained from Scratch	46.07	34.17	21.38	12.82
Debiasing Model Approach	48.73	34.43	22.36	14.21

Table 5: Performance comparison of models across datasets.

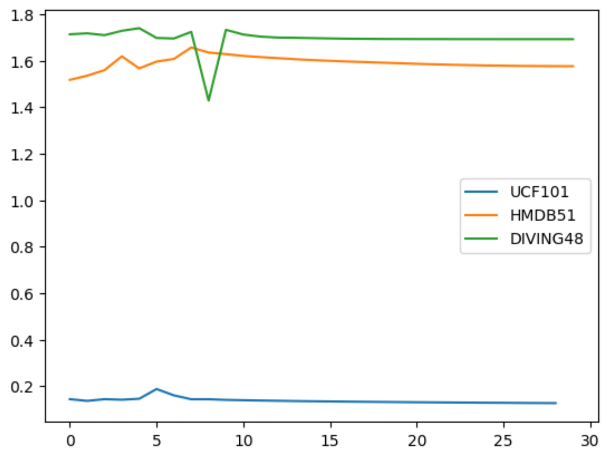
In addition to this, the work can be further refined by training the models for more epochs to achieve higher and more stable accuracies. This would ensure the models are fully optimized and provide results that are suitable for real-world applications. Furthermore, the scope of this research can be extended to various other applications of convolutional neural networks (CNNs). For instance, in the domain of autonomous vehicles, CNNs can be utilized for vehicle recognition systems that focus on critical features like shape, size, and motion patterns, while avoiding reliance on irrelevant attributes such as color or background environment. The debiasing framework could also be explored in agricultural applications, such as plant disease detection, where the model should learn features specific to diseases rather than being misled by factors like lighting conditions, soil type, or background foliage. These applications illustrate the vast potential for extending and adapting the core ideas of this work to other impactful areas. CNNs are still and at large impacting the realm of deep learning due to theri major ability of feature extractions and techniques like this further help in increasing the efficient of the CNNs and deep learning in general.

6 Contributions

All of us contributed equally to this project right from since the beginning, including brainstorming. We worked on this project in two phases and the contribution summary of each member



(a) Validation loss of model without debiasing on transfer learning tasks



(b) Validation loss of model with debiasing on transfer learning tasks

Figure 8: Comparison of validation loss with and without debiasing on transfer learning tasks.

in both the phases is as follows:

6.1 Phase 1

- Sai worked on the Dataset Preparation and Frame extraction part. Initially, we all contributed in the frame extraction, since it was computationally expensive to download and extract frames from such huge datasets, especially Mini Kinetics dataset.
- Sai worked on Masking of the videos for our debiasing model.
- All of us worked together in understanding the research paper, especially the loss functions.
- Swabhi and Vishnu worked together on creating the baseline model and running different experiments for transfer learning.
- Swabhi worked on creating the FrameDataset class to handle data loading, preprocessing, and augmentation for training and evaluation.
- Vishnu Generated pseudo-scene labels using the Places365 model and managed data preparation pipelines for the Mini-Kinetics dataset.

6.2 Phase 2

- Vishnu worked on creating the structure for debiasing model, implementing the novel losses addressed by the paper.
- Swabhi conducted detailed literature review and identified key methodologies from the reference paper, and worked on the transfer learning for the debiasing model.
- Sai analyzed the results and compared baseline and debiasing models' performances across datasets.

All of us equally contributed in writing this final project report.

Acknowledgments

We appreciate the work of the original authors of this paper and also convey our sincere thanks to Dr. Liu for offering this course. We inform the readers that OpenAI's generative artificial intelligence was used to improve our writing conciseness.

References

[1] Choi, J., Gao, C., Messou, J. C. E., and Huang, J.-B. Why can't i dance in the mall? learning to mitigate scene bias in action recognition. In *Advances in Neural Information Processing Systems* (2019), vol. 32.

[2] Ganin, Y., and Lempitsky, V. Domain-adversarial training of neural networks. In *International Conference on Machine Learning (ICML)* (2015), pp. 1180–1189.

[3] Hara, K., Kataoka, H., and Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 6546–6555.

[4] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[5] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., and Natsev, P. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).

[6] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)* (2011).

[7] Li, Y., Wang, Z., Wang, J., Wu, Y., Qian, C., Yang, M.-H., and Cottrell, G. W. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 513–528.

[8] Soomro, K., Zamir, A. R., and Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).