CS230

# Activity Recognition with Still Images and Video

**Category: Computer Vision**

**Wendell Hom**
SCPD
Stanford University
froi@stanford.edu

## Abstract

Human Activity Recognition (HAR) is an important and well studied topic in Computer Vision. Here, we explore some architectures that have been used to perform action recognition on both still images and videos.

For still images, we used several well known architectures to fine tune on the Stanford40 dataset and achieved 84.1% accuracy with the Inception-ResNet-v2 model. We then implemented a paper [5] that includes a Human Localization and an Action Classification branch. This model gave us an accuracy of 84.7% on the validation set, however, it did not generalize well to inference in the wild.

For videos we evaluated a 2D CNN, a shallow 3D CNN, and a deep 3D CNN on a 15-category subset, a 27-category subset, and then the full 600-category Kinetics dataset. For the 15-category subset, the 2D CNN fine tuned on the Stanford40 dataset gave us the best results at 59% accuracy. For the 27-category subset and the full 600 category dataset, only the shallow and deep 3D models were evaluated. For the 27-category subset, the shallow 3D CNN gave better results with an accuracy of 43.7%. For the 600-category dataset, the shallow 3D CNN had a higher accuracy at 35.2%. Our work is publicly available and can be downloaded from our github repo.[1]

## 1 Introduction

Action Recognition has many important applications including security surveillance, healthcare, and pedestrian monitoring for autonomous vehicles. With advances in both computing power and research in Deep Learning, these applications have made many breakthroughs in recent years.

Our work here aims to evaluate a number of neural networks to see how well they can perform HAR on the Stanford40 dataset for images, and the Kinetics dataset for videos. Given an image, or a video, we will evaluate whether the models can correctly classify the activity.

## 2 Relevant Work

A number of 2D CNN architectures are available for image classification tasks with weights pre-trained on ImageNet. These include the architectures we used: **VGG16** and **VGG19**, **ResNet-50**,

[1]https://github.com/wendell-hom/human_action_recognition

**MobileNet-v2**, **Inception-ResNet-v2**, and **Inception-v3** which we leverage to fine-tune on the Stanford40 set for action classes. Additionally, work has been done by Lu *et el* [5], which makes use of additional features such as a Human mask to help the neural network pay attention to relevant parts of an image, specifically those pixels where the human subject is located.

On the other hand, while 3D CNN architectures have also been widely studied, pre-trained models are not built into the Keras API that we used. Instead, we opted to implement the shallow **C3D** CNN network described in [6] as well as a 3D CNN version of **ResNeXt-101** [7] which was used in the work of Hara *et el* in their research [1] to see if Spatiotemporal 3D CNN's can retrace the success of 2D CNN's on ImageNet.

## 3 Dataset

For still image HAR, the Stanford40 Action dataset [4] was used which contains 40 different human activity classes with 180-300 images per action class. Action categories include "brushing teeth", "playing guitar", "jumping", among others. Additionally, this dataset contains annotations for the bounding boxes of the human relevant to the action and is used in one of the implementations to create a human mask. Due to relatively small size of the dataset, we decided to go with at 70/30 train-dev split.

For video HAR, the Kinetics-400 dataset [2] was used. The Kinetics-400 dataset contains a collection of videos that classify 400 different human actions, with at least 400 video clips for each activity. Since the provided test set was not labeled, only the test and validation sets were used. Initially, a 15-category subset which overlapped between the Stanford40 and Kinetics dataset were used to compare video HAR using a 2D CNN, with a shallow 3D CNN and deep 3D CNN trained from scratch using the 15 categories from Kinetics. Next a subset of 27 categories were used to train the shallow 3D CNN and deep 3D CNN and evaluated. Finally, as an extra challenge, the Kinetics-600 dataset [8] was downloaded and an attempt to train our models on such a large dataset was attempted.

### 3.1 Data Preprocessing

#### 3.1.1 Image Preprocessing and Augmentation

- images were rescaled in the spatial dimension to match the input shape of the model
- for the 2-branch model, a human mask was generated for each image based on bounding box coordinates provided in the Stanford40 dataset. This ground truth human mask was rescaled to the same dimensions as the output of the human localization branch which was 17x17x1.
- for augmentation, horizontal flipping, a rotation range of 10 degrees, and a zoom range between [0.9, 1.1] was used with the exception of the 2-branch model which makes use of bounding boxes and precluded the use of the Keras image augmentation since our human mask would no longer be accurate.

#### 3.1.2 Video Preprocessing and Augmentation

- it takes extensive computational resources to train a 3D CNN. To reduce loading time from compressed mp4 video format, the video frames were first extracted using OpenCV and then written out to compressed hdf5 format with one hdf5 per category per test and validation set. Dataframes were used to help retrieve the corresponding video frames for each video. The disk usage for Kinetics-600 required 4 TB of disk space. SSDs were needed to ensure fast read times.
- during preprocessing, video frames from each video were resized and stored in the hdf5 files as Lx171x128x3 numpy arrays, where L is the total number of frames of the video. During video augmentation, 16 random consecutive frames were picked with a frame rate in the range of [1, 5] and cropped to a spatial dimension of 112x112 pixels. All 3 color channels were used.

## 4 Methods

### 4.1 Still Image HAR

For still images, we used transfer learning on a number of models pre-trained on ImageNet weights, replacing the final layer with a 40-way softmax output layer. The CNN base layers were frozen.

Afterwards, we picked the Inception-ResNet-v2 model and implemented an architecture where the model has two branches as shown in Figure 1. The first branch is the Human Localization branch which is used to predict the human heat mask. The second branch is the same action classification branch we have in our other image classification models. The output of the action branch uses the softmax loss function while the output of the human localization branch uses a mean squared error loss function computed from the predicted human mask, $M^*$, along with the ground truth mask, $M^{gt}$, as seen in Equation 1. Both losses were weighted equally. For this model, per the clarification of the original authors, all layers are trainable including the CNN base layers.

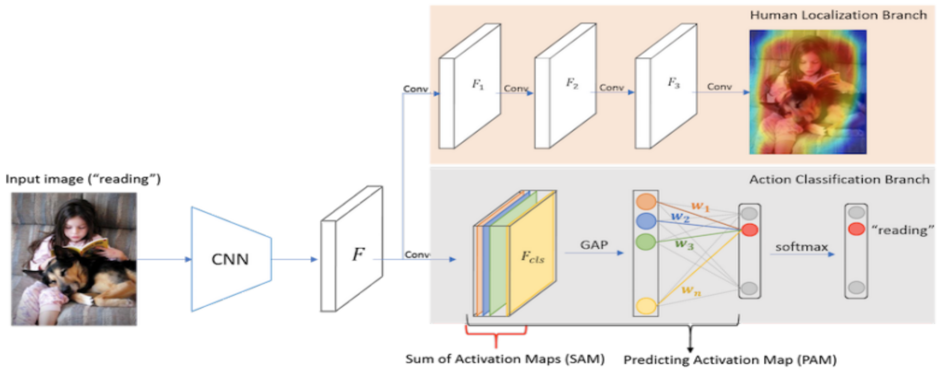

Human Localization Branch
Input image ("reading")
CNN
F
Conv
$F_1$
$F_2$
$F_3$
Action Classification Branch
$F_{cls}$
GAP
$w_1$
$w_2$
$w_3$
$w_n$
softmax
"reading"
Sum of Activation Maps (SAM)
Predicting Activation Map (PAM)

Figure 1: Loss Guided Activation for Action Recognition in Still Images

$$L_{cls} = -\log\left(\frac{\exp\left(S_{c^{gt}}\right)}{\sum_c \exp\left(S_c\right)}\right); \qquad L_{mask} = \left\|M^{gt} - M^*\right\|_2^2 \tag{1}$$

All models mentioned above included a Global Average Pooling layer to make it easy to visualize the Class Activity Mapping later. We trained these models using the Adam optimizer with an initial learning rate of 0.0001, and reduced it to 0.00005 and then 0.00001 as the training loss saturated.

### 4.2 Video

Our first method for video activity recognition is to take the 2D Inception-ResNet-v2 model we fine-tuned with the Stanford40 action dataset and apply it to each frame of the input video individually. We then average the values across all frames and select the class with the highest score. For this method we chose 15 categories from Kinetics that overlapped with the Stanford40 dataset.

The next model we used is C3D which is the shallow 3D CNN depicted in Figure 2. For the deep 3D CNN model, we used a 3D version of the ResNeXt-101 model with a cardinality of 32. For testing, we split a video into non-overlapping windows of 16 frames and classified each 16-frame window individually. We then averaged the values among all the windows and select the class with the highest score. If the final window has less than 16 frames, we drop it.

When training these 3D CNN models on the 15-category and 27-category subset of Kinetics, we used the Adam Optimizer with an initial learning rate of .0001 and reduced it to .00005 and then .00001 as the training loss saturated. For training the 600-category Kinetics dataset, we closely monitored the training and experimented with different learning rates in the range of [0.00005, 0.0001] when it looks like training was not making progress.

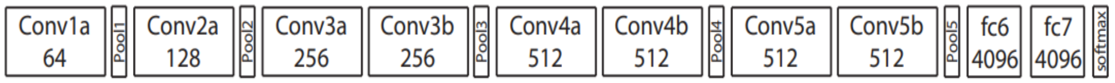


Figure 2: C3D architecture. C3D net has 8 convolutions, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are 3x3x3 with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from pool1 to pool5. All pooling kernels are 2x2x2, except for pool1 is 1x2x2. Each fully connected layer has 4096 output units.

## 5 Results

For our metric, we chose to use accuracy because the number of images in our datasets were relatively well balanced and the papers we referred to used accuracy as the metric.

### 5.1 Image HAR

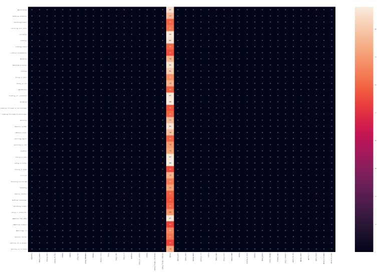
For Image HAR, our findings are shown in Table 1.

| **Method** | **Pixels** | **Num Parameters** | **Accuracy** |
|---|---|---|---|
| MobileNet-v2 | 224 | 14.1 M | 65.1% |
| VGG-16 | 224 / 500 | 19.5 M | 56.6% / 59.6% |
| VGG-19 | 224 / 500 | 24.8 M | 55.2% / 57.8% |
| ResNet-50 | 224 / 500 | 42.5 M | 2.7% / 1.9% |
| Inception-ResNet-v2 | 224 / 500 | 68.5 M | 71.6% / 84.1% |
| Inception-v3 | 224 / 500 | 40.7 M | 64.5% / 83.7% |
| Loss Guided Act. | 500 | 75.9 M | 84.7% |

Table 1: Model Size and Accuracy

From our runs, the MobileNet-v2 model gave the best bang for the buck with a smaller model size, signficantly faster training times, and a decent accuracy even when the input pixel resolution was limited to 224x224.

The ResNet-50 model was interesting as it seemed to have memorized the training set without generalizing; it achieved very high training accuracy but the validation accuracy was a dismal 1.9%-2.7%. When viewing the confusion matrix in Fig. 3a, it turns out that the network had classified all images as "Phoning" (or another class depending on the run) for images that were not in the training set.


(a) ResNet-50

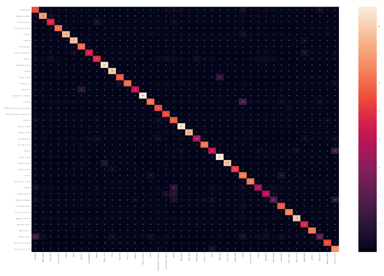

(b) Inception-ResNet-v2

Figure 3: Confusion Matrix as a heatmap for two models. The ResNet-50 model failed to generalize

The Loss Guided Activation model achieved the highest validation accuracy, but it performed poorly on images downloaded from the web so it seemed to have generalized poorly as well. Between Inception-ResNet-v2 and Inception-v3, our results had a slightly higher accuracy for Inception-ResNet-v2 so we used it for the following work in categorizing Video HAR. Additionally, we ran the training on input resolutions of 224x224 as well as 500x500 for a few models. Looking at Table 1, it appears that increasing the resolution from 224 to 500 helped models that had higher modeling capacity such as Inception-ResNet-v2 and Inception-v3, but didn't make much of a difference for VGG-16 and VGG-19.

### 5.2 Video HAR

| Method | Parameters | Acc. 15-class | Acc. 27-class | Acc. 600-class |
|---|---|---|---|---|
| 2D CNN | 68.5 M | 59.0% | - | - |
| C3D | 63.4 M | 42.5% | 43.7% | 35.2% |
| ResNeXt-101 | 47.8 M | 41.1% | 37.6% | 24.4% |

Table 2: Model Size and Accuracy

For videos, the 2D Inception-ResNet-v2 CNN we fine-tuned using Stanford 40 action dataset gave us the best performance at 59.0% accuracy for a 15-category subset that was common between Stanford 40 and Kinetics dataset. This gives us a hint that training 3D CNNs is not an easy task.

Contrary to our expectations, the accuracy rate of the 3D CNNs for the 27-category subset and the full 600-category Kinetics dataset was even lower. However, this is mainly because it becomes much more difficult to train these models due to the high computation resources required, making it very difficult to experiment with various hyperparameters.

One other surprising finding was that while the ResNeXt-101 has fewer parameters than the shallow C3D model, it required a lot more memory. We believe this is largely due to its depth, since the network has to cache values calculated during forward propagation for use in the backward propagation step.

## 6 Discussion and Future Work

### 6.1 Image HAR

For the Loss Guided Activation 2-branch model, future work can focus on why it generalized poorly to new images. We may want to try adding bounding boxes to the input images that are tested to see if it makes any difference. Additionally, this was the only model where we were not able to use image augmentation so another avenue for improvement may be to implement our own custom image augmentation or leverage existing repos or packages that may provide this feature. For the ResNet-50 model, we could try choosing a lower learning rate to see if that helps.

### 6.2 Video HAR

Due to compute and time resources, as well as the difficulties of training a 3D CNN from scratch, it was much easier to fine-tune a 2D CNN pre-trained on ImageNet. For our future work, we will look at exploring more sophisticated methods for hyperparameter tuning in hopes that we can get better results for our 3D models. Furthermore, it seems that the work by Hara *et al* and others that publish state-of-the-art papers prefer the use of SGD over Adam because SGD generalizes better so we may want to try using SGD instead. Another difficulty with training 3D models, especially the deep 3D CNN models, is that it takes too long to train. One direction that can be explored is to leverage multi-gpu training to reduce the turn-around-time in order to better gauge and tune our hyperparameters to get better results.

## References

[1] K. Hara, H. Kataoka, Y. Satoh. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? In CVPR, 2018.

[2] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, A. Zisserman. The Kinetics Human Action Video Dataset. In arXiv:1705.06950, 2017.

[3] A. Bevilacqua, K. MacDonald, A. Rangarej, V. Widjaya, B. Caulfield, T. Kechadi. Human Activity Recognition with Convolutional Neural Networks. In arXiv:1906.01935, 2019.

[4] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei. Human Action Recognition by Learning Bases of Action Attributes and Parts. In International Conference on Computer Vision (ICCV), Barcelona, Spain. November 6-13, 2011.

[5] L. Liu, R. T. Tan, S. You. Loss Guided Activation for Action Recognition in Still Images. In arXiv:1812.04194, 2018.

[6] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features With 3d Convolutional Networks. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 4489–4497. IEEE, 2015.

[7] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1492–1500, 2017.

[8] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier. A Short Note about Kinetics-600. In CVPR, 2018.

[9] N. Shirish Keskar, R. Socher. Improving Generalization Performance by Switching from Adam to SGD. In arXiv:1712.07628, 2017.