Human Activity Recognition Using LSTM

Aya Ismail [1] **Khoi Pham** [1] **Kaan Elgin** [1] **Karan Deep Kaur** [1] **Shambhavi Kumar** [1]

Abstract

Human activity recognition is a traditional task in Computer Vision which is based on classifying different actions that humans perform in videos. We present a model based on neural networks, in particular CNN (Convolutional Neural Network) and LSTM (Long Short-term Memory Network), which achieves this task. We use temporal features in our model which are based on differences between consecutive frames of videos.

1. Introduction

Activity recognition is an active area of Computer Vision which has many applications including robot control, human computer interaction, and visual surveillance. We present a model for performing activity recognition which is based on CNN and LSTM models. This model combines the feature extraction capabilities of CNN with time-series predictions of LSTM. Our baseline is SVM (Support Vector Machines) with bag-of-words model which has been widely used in the context.

2. Background

2.1. SVM with Bag-of-Words Model

Bag-of-words (BoW) model is widely used in images as well as text-based applications. In this context, words represent visual features. The basic approach is the following: After computing the features, they are clustered using k-means clustering algorithm. The cluster centers represent the visual words. After that, to count words in each image, histograms are computed from the cluster assignments of pixels. Those histograms are the features for the classifier where the dimension of feature space is equal to the number of clusters. In literature, SVM (Support Vector Machines) classifier is widely used for that purpose as in (Csurka et al., 2004).

2.2. CNN

CNN has been proven to be successful in image tasks in recent years, such as image classification and detection. There have been studies regarding visualization and interpretation of the learned CNN filters, and it has been shown by (Zeiler & Fergus, 2013) that CNN effectively learns to detect basic features such as edges, corners, circles, or even high level features such as car wheels and humans.

Inspired by the competitive performance on images, CNN is now also being used to tackle video understanding problems, e.g. activity recognition and object tracking. In our case, we should design our CNN such that it is able to learn features that evolve over time. A recent work tackling activity recognition problem by (Karpathy et al., 2014) shows that CNN with 3D filter can be employed to learn spatio-temporal feature, that is, feature in both the spatial and temporal domain. Inspired by this method, we explore training our own CNN models with different architectures to study the capabilities of CNN.

2.3. LSTM

The problem with traditional recurrent neural networks is that it can be difficult to train them to learn long-term dynamics, likely due in part to the vanishing and exploding gradients problem. A popular choice to solve this issue is the Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). We use the LSTM unit shown in figure 1.

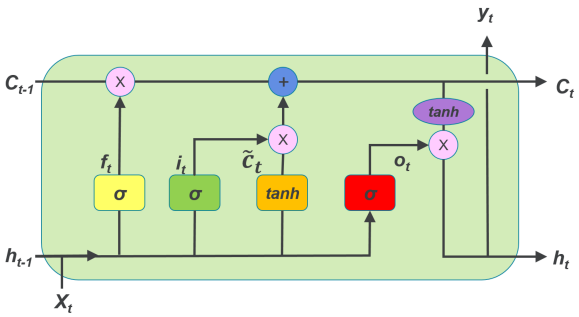

Figure 1. Long Short-term Memory Neural Network

LSTMs provide a solution by incorporating memory units that explicitly allow the network to learn when to forget previous hidden states and when to update hidden states given new information. The state updates satisfy the following operations:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\tilde{\mathbf{c_t}} &= \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c_t}} \\
\mathbf{h}_t &= \tanh(\mathbf{o}_t \odot \mathbf{c}_t)
\end{aligned}
\tag{1}
$$

Here σ is the logistic sigmoid function; $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c$ are recurrent weight matrices; $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c$ are the bias terms. In addition to a hidden unit $\mathbf{h}_t$, the LSTM includes an input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, output gate $\mathbf{o}_t$, input modulation gate $\tilde{\mathbf{c_t}}$, and memory cell $\mathbf{c}_t$.

3. Model

3.1. Baseline Model

We took SVM with bag-of-words model as our baseline. We used two types of features and sampling techniques. The features we used are SIFT (Scale-invariant Feature Transform) and optical flow.

SIFT is a spatial feature which represents pixels according to the orientations of their neighborhood and it does not change when scale of the image changes. It takes a 16x16 neighborhood for each pixel and divides it into 16 blocks. Then for each block, it computes an orientation histogram with 8 bins. So the features are 128 dimensional at the end. The drawback of SIFT is it does not consider the frames in a video as a sequence since it is not a temporal feature.

Unlike SIFT, optical flow is a temporal feature which computes the velocities of the pixels by tracking them throughout the frames. It is a two dimensional feature which represents the velocities in x and y directions. For computing optical flow features we used Farneback algorithm (Farnebäck, 2003).

For sampling the pixels, we used keypoint sampling and dense sampling. For keypoint sampling, we detected the corner points and computed the features for them. The other approach we tried was dense sampling which samples points by a regular interval. In that case, the amount of data becomes higher.

3.2. Intermediate Model

We explored multiple CNN architectures, namely 2D-CNN, 3D-CNN, and 3D-Flow-CNN. The differences between them are the types of input (raw frame or raw frame with optical flow) and the kinds of filter (2D or 3D) that are used at the convolutional layer. We inherited the idea from (Latah, 2017) to design our CNN models' architectures. We wanted to study how the 3D convolutional layer is able to extract features in the temporal domain and outperforms the 2D version. We also wanted to study whether employing the optical flow information will help to improve performance.

3.2.1. 2D CNN ON RAW IMAGE

In order to evaluate CNN's capability in extracting useful features from static images, we train a CNN with 2D filter as in image classification task. Our architecture is described as follows with shorthand notation: $Conv(16,5)$-BN-$Relu$-$Pool$-$Drop(0.5)$-$Conv(32,3)$-BN-$Relu$-$Pool$-$Drop(0.5)$-$Conv(64,3)$-BN-$Relu$-$Pool$-$Drop(0.5)$-$FC(128)$-$Drop(0.5)$, where $Conv(d,f)$ is a convolutional layer with d filters and filter size $f \times f$, BN is a batch normalization layer, P is a pooling layer with filter size 2×2 and stride 2, and $FC(n)$ is a fully connected layer with n neurons. The output after the last fully connected layer is connected to a softmax classifier with 6 classes corresponding to 6 activities that we want to classify. The architecture of this model is presented in Figure 2.

This model is trained on video frame level as in image tasks. To make classification for a video, we classify all its frames, then average the probabilities of all classes across the frames and choose the class with the highest probability.

3.2.2. 3D CNN ON RAW IMAGE

This is a modified version of the above 2D CNN. Instead of 2D filter, we use 3D filter that extends to the time domain. An input instance to this model is not a single frame but a block of 15 consecutive frames (Figure 2). Utilizing 3D filter across a time window on consecutive frames allows the model to detect both visual and motion information at the same time.

However, there is one glitch in this idea: a 3D filter does not just learn motion features, but actually it learns to detect special features in the correlation between frames in a block, including motion features. For example, in the image classification task, even if we are to permute the order of channels in an image, the well-known existing CNN architectures will still be able to learn visual features and make good classification result. Similarly, in our activity classification problem, even if we permute the order of the frames in a block, and keep that permutation consistently across all training instances, our model still achieves the same performance. Therefore, it comes to us that using optical flow as an additional input to our 3D CNN model might be a good idea because it enforces the model to focus on the motion features instead of extracting features from the correlation between all frames.

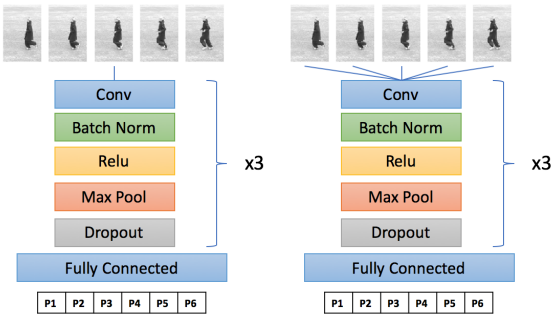

Figure 2. (Left) 2D CNN on single frame (Right) 3D CNN on block of consecutive frames.

3.2.3. 3D CNN ON RAW IMAGE AND OPTICAL FLOW

This model has the same architecture as the 3D CNN described in the previous section, however, it duplicates the 3D CNN by 3 times corresponding to 3 streams of inputs: raw frame, optical flow in the horizontal direction, and optical flow in the vertical direction (Figure 3). This network is able to learn useful features regarding the motion.

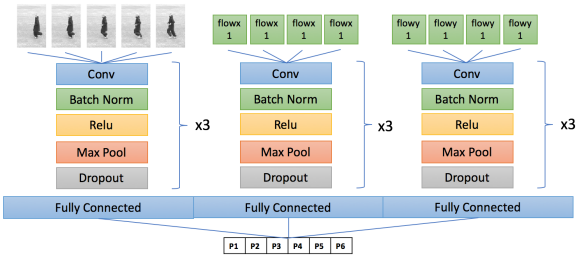

Figure 3. 3D CNN on block of consecutive frames and optical flows.

3.3. Final Model

Our final model was a combination of LSTM and CNN, in which CNN's function was feature extraction and LSTM was responsible of classification. This method was introduced by (Donahue et al., 2015). We used the third CNN architecture described in Section 3.2. During preprocessing, frames were resized to 80 x 60. We only considered frames that contain humans in them. The outputs of CNN were 128 dimensional feature vectors which were then grouped and fed into the LSTM as a 5 timestep per sample. For classification a shallow 2 layer LSTM with 150 neurons, each connected to a softmax layer, was used. The loss function was cross-entropy of the last timestep in LSTM as shown in the equation below:

$$-\sum_{i}^{M} L_i \log(S_i)$$

where M is the number of classes, L represents the one-hot encoding of the true class and S is the probability distribution taken from softmax layer.
During testing, to produce a single label prediction for an entire video clip, we averaged the label probabilities, which are the outputs of the softmax layer, across all frames and choose the most probable label. The complete model is shown in figure 4.

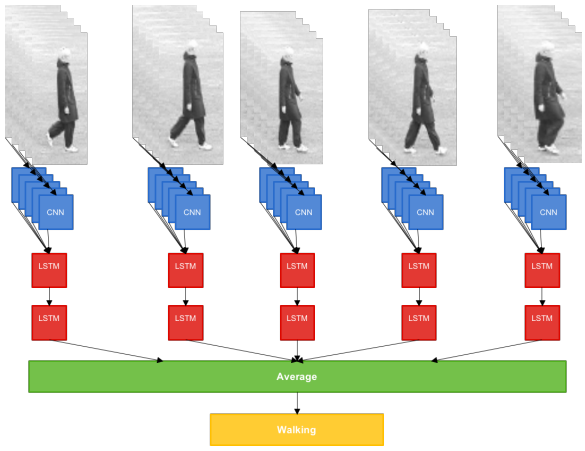

Figure 4. CNN-LSTM Model

We also investigated other CNN models including (Krizhevsky et al., 2012) and (Simonyan & Zisserman, 2014); which are pre-trained networks on the 1.2M image (Russakovsky et al., 2015) classification training subset of the ImageNet (Deng et al., 2009) dataset. Although our 3D-CNN was only trained using 4885 training samples it gave better result as shown in section 4. This is because our CNN was trained for only this specific task while (Deng et al., 2009) dataset contains images like the ones in Figure 5 and 6.

Figure 5. ImageNet Training example

Figure 6. ImageNet Training example

4. Experiments

4.1. Data Description

The dataset consists of video files. A video consists of a human doing one of the six following activities: Walking, boxing, jogging, running, hand clapping, hand waving. There are 25 different human figures who are doing the activities multiple times in both indoor and outdoor environments. All sequences were taken by a static camera with 25 fps frame rate and have a length of four seconds in average. For our experiments, we took 191, 192, and 216 videos for training, development, and testing respectively where the splitting is provided by the KTH Institute (Laptev & Caputo, 2005).

4.2. Baseline Experiments

The maximum accuracy we got from SIFT is 50.93% which is not much high. However since optical flow is a temporal feature, we got higher accuracy using that. Figure 7 shows the accuracy achieved with optical flow.

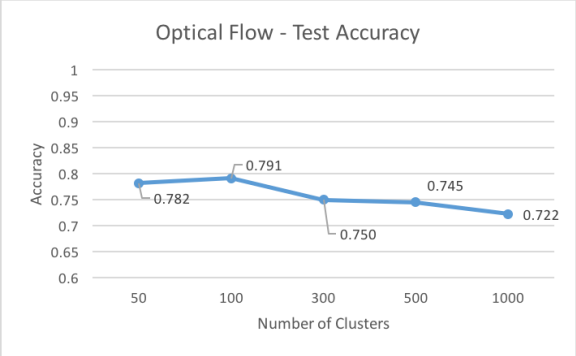

Figure 7. Baseline Accuracy using Optical Flow

These features were extracted using OpenCV library.

4.3. Intermediate Experiments

The CNN models were implemented using Keras library of Python. We examined three models: 2D CNN on single frame, 3D CNN on blocks of raw frames, and 3D CNN on blocks of raw frames and flows. The test accuracy of the models are shown in Table 1.

Model	Accuracy on Test Set
2D CNN - Single Frame	0.736
3D CNN - Block of Frames	0.787
3D CNN - Block Frames + Flows	0.884

Table 1. Accuracy using CNN models.

Test accuracy of the 2D CNN model outperforms the baseline method that uses SIFT feature. This shows that the CNN is able to extract more useful and descriptive features than SIFT. Furthermore, when we extend to the time domain and employ a CNN with 3D filter, the accuracy goes up to 0.787, which proves that the 3D CNN succeeds in learning with the temporal features. The 3D CNN that was trained on both raw frames and flow features outperforms the previous CNN models.

4.4. Final Experiments

The LSTM model was implemented using Tensorflow library of Python. The experiments were carried for different input parameters. We chose the best performing models for each combination of input parameters and compared them. The following graphs show the accuracy of the final model with respect to individual parameters.

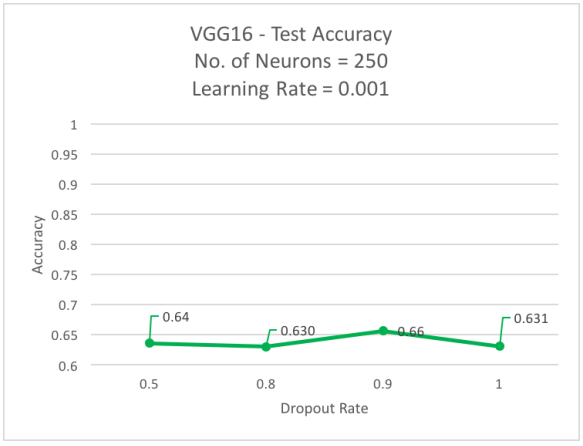

Figure 8. VGG16+LSTM Accuracy vs. Dropout Rate

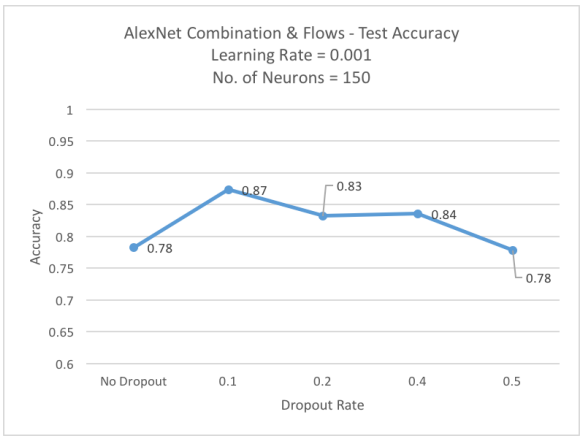

Figure 9. AlexNet+LSTM Accuracy vs. Dropout Rate

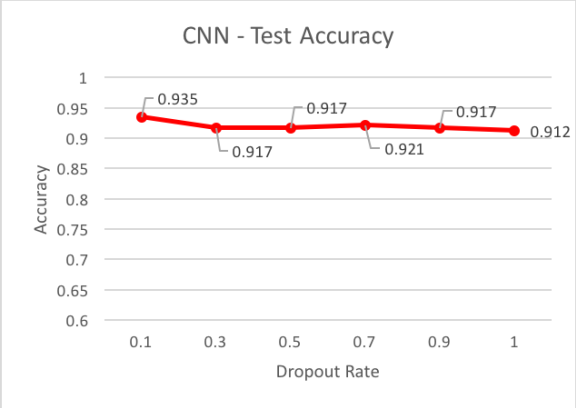

Figure 10. Accuracy of final model with our CNN vs. Dropout Rate

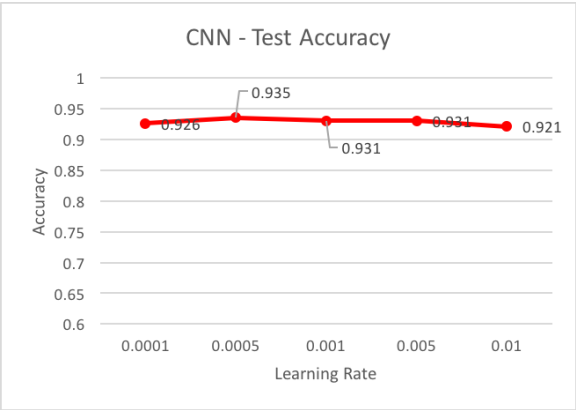

Figure 11. Accuracy of final model with our CNN vs. Learning Rate

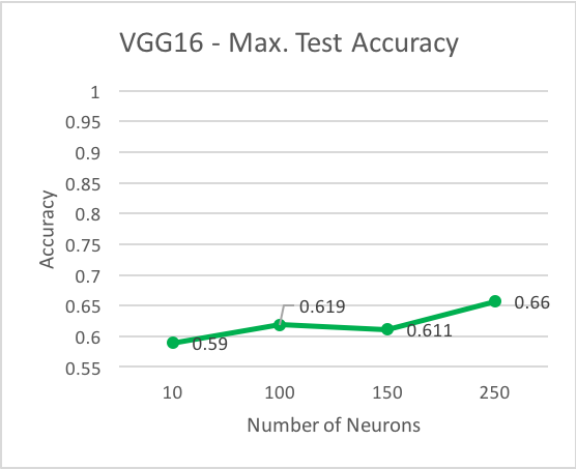

Figure 12. VGG16+LSTM Accuracy vs. Number of Neurons

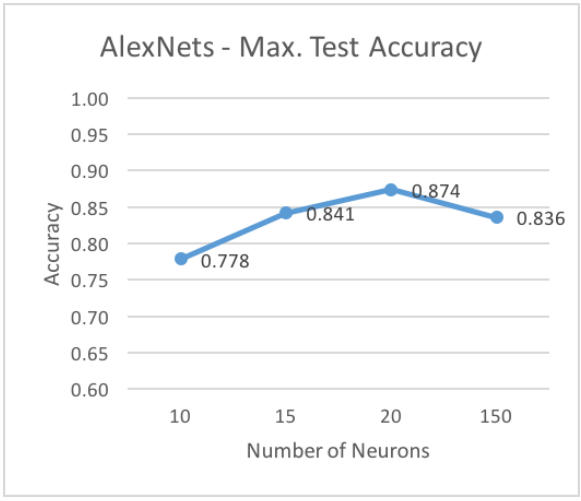

Figure 13. AlexNet+LSTM Accuracy vs. Number of Neurons

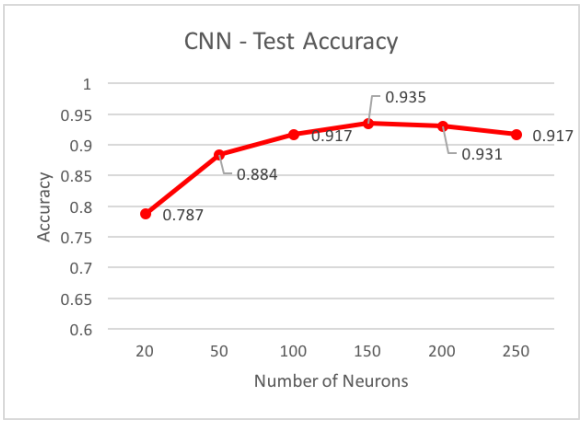

Figure 14. Accuracy of final model with our CNN vs. Number of Neurons

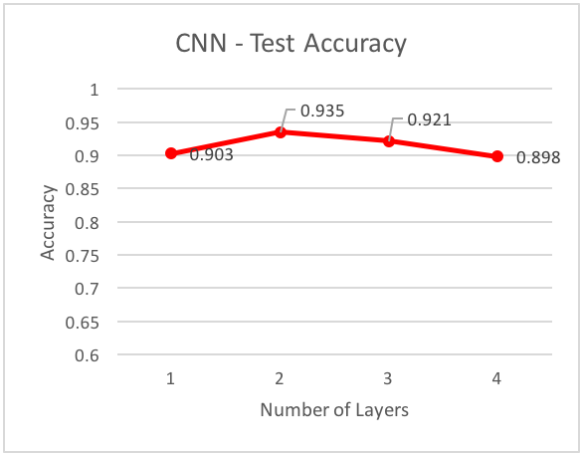

Figure 15. Accuracy of final model with our CNN vs. Number of Layers

COMPARISON OF CNNS

Upon performing several experiments, the maximum test accuracy for VGG16 CNN was 0.656. Whereas for AlexNet, the accuracy was 0.873. Finally, CNN model trained by us had the best performance overall and gave 0.935 accuracy. The optimal parameters are learning rate = 0.0005, dropout rate = 0.1, number of layers = 2, and number of neurons = 150.

4.5. Results

AlexNet and VGG models were pre-trained, which means that they were also trained on data apart from the videos. The CNN we built was solely trained on the relevant video dataset as input. The maximum test accuracy for this model was highest at 0.935. The final confusion matrix is the following with the classes 'Walking', 'Jogging', 'Running', 'Boxing', 'Handwaving', 'Handclapping' respectively:

$$\begin{bmatrix} 36 & 0 & 0 & 0 & 0 & 0 \\ 1 & 32 & 3 & 0 & 0 & 0 \\ 0 & 4 & 32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 36 & 0 & 0 \\ 0 & 0 & 0 & 3 & 30 & 3 \\ 0 & 0 & 0 & 0 & 0 & 36 \end{bmatrix}$$

As seen in the matrix, there is some confusion between running and jogging; and that was one reason to use optical flow feature which provides an estimate of velocity. Also sometimes handwaving is classified as boxing or handclapping.

5. Conclusion and Discussion

The motivation behind this project was not only to build an accurate prediction model, but also to explore use of CNN and LSTM models in video classification. In conclusion, we provided a model based on CNN and LSTM which outperforms simpler models as bag-of-words and single CNN.

6. Contributions

- Aya Ismail did the LSTM part.
- Khoi Pham did the CNN part and had contributions to the baseline.
- Kaan Elgin did the feature extraction and baseline, and had contributions to the LSTM model.
- Karan Deep Kaur had contributions to the feature extraction part.
- Shambhavi Kumar had contributions to the LSTM experiments.

In addition, source code is available in the following Github repository: https://github.com/turankaanelgin/Action-Recognition.

References

Csurka, Gabriella, Dance, Christopher, Fan, Lixin, Willamowski, Jutta, and Bray, Cédric. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pp. 1–2. Prague, 2004.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

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

Farnebäck, Gunnar. Two-frame motion estimation based on polynomial expansion. *Image analysis*, pp. 363–370, 2003.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Karpathy, Andrej, Toderici, George, Shetty, Sanketh, Leung, Thomas, Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Laptev, Ivan and Caputo, Barbara. Recognition of human actions. 2005.

Latah, Majd. Human action recognition using support vector machines and 3d convolutional neural networks. *International Journal of Advances in Intelligent Informatics*, 3(1):47–55, 2017.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Zeiler, Matthew D. and Fergus, Rob. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL http://arxiv.org/abs/1311.2901.