CS230

# Gesture Classification From RGB and RGB-D Video Using LSTMs with Optical Flow Features

**Kenny Leung***
Department of Computer Science
Stanford University
kenleung@stanford.edu

## Abstract

Optical flow captures the positional displacement of each pixel from one frame to the next. This study evaluates an approach to video gesture classification that performs feature extraction using optical flow. The features are extracted from temporally-adjacent pairwise frames of the video; the resulting sequence is then fed into an LSTM classifier. This model is evaluated on the ChaLearn Looking at People Isolated Gesture Recognition challenge (ICPR '16). An optical-flow-based classifier achieves 2.16% validation accuracy and 2.30% test accuracy on a dataset consisting of 249 distinct gestures and 35K videos. This is shown to perform worse than the baseline RGB-RGBD classifier, which achieves 7.24% test accuracy.

## 1 Introduction

Video gesture classification involves determining, given video of an actor gesticulating, which predefined action s/he intends to portray from a limited set of possible actions. A general gesture classifier may help further the development of sign language recognition by capturing latent features in sequences of images which are useful for discriminating between different bodily gestures. The video gesture classification task is challenging due to the inherent physical and temporal variations of gestures performed across different actors, as well as noisy backgrounds and the slightly variable differences in perspective, such as lighting conditions and distance between the camera and the actor.

This study investigates whether the inclusion of optical flow features improves the gesture classification task. Optical flow [1] captures the positional displacement of each pixel in a video frame, relative to the last frame. Using pixel displacement as a feature is a good feature candidate for a gesture classifier. Firstly, because the video samples in the dataset are captured from a fixed frame of reference, background pixels remain static throughout the videos; thus, optical flow implicitly performs foreground detection of the actor. Secondly, optical flow measures the relative speed of movement of individual pixels. This may be useful in cases of self-occlusion, where one body part is partially obscured by the other. Because optical flow considers individual pixels as part of the same object across independent frames, it may be able to determine that a hand is moving across the frame (even if it becomes partially occluded), as opposed to treating each pixel as an independent data point.

*In collaboration with Ying Hang Seah (yinghang@stanford.edu) and Hiroshi Mendoza (hmendoza@stanford.edu).

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Figure 1: Kinect RGB-D (left) and RGB (right) video frames depicting an actor performing a handshake gesture, occurring adjacently at a video sampled at 10 fps.

## 2 Background and Related Work

There are a variety of techniques and model architectures that have been employed to address the gesture classification task. A common approach is to perform data pre-processing on each frame from the RGB and RGB-D videos using Histogram of Gradients (HOG) [2], Pose Estimation [3] and Saliency Theory [4]. The processed data are often combined with the raw RGB and RGB-D images, which are then stacked temporally and fed to a 3D-CNN to derive the output classification.

### 2.1 ICPR '16 Isolated Gesture Recognition

A team from Xidian University achieved state-of-the-art accuracy of 56.9% as part of the classification challenge associated with our dataset. They employ 3D convolutional neural networks and saliency feature extraction [5]. A saliency video [4] is created from the RGB video and its features are extracted using a 3D-CNN. The feature vectors are then concatenated and passed through a SVM classifier.

Another team from the University of Wollongong achieved a test accuracy of 55.57% on this challenge using Dynamic Depth Images, Dynamic Depth Normal Images and Dynamic Depth Motion Normal Images for the feature extraction by only using the depth data. [2] Features extracted from HOG are then passed through Bidirectional Rank Polling and 2D-CNN to create a sequence of score vectors. The score vectors are fused and normalized to create a sequence of final score vectors.

### 2.2 Two Stream Convolutional Networks for Action Recognition

This project is largely inspired by the work of Simonyan and Zisserman [6], which demonstrated that high gesture classification accuracy can be achieved using a convolutional neural network trained using two-stream (1) RGB and (2) optical flow input. Lucas-Kanade optical flow [7] is used to compute the horizontal and vertical displacement of each pixel within a frame, conditioned on the previous frame. Optical flow features measure velocity of objects across sequential frames; it need not indirectly learn this information by recurrently processing sequential frames.

## 3 Dataset and Features

Our classifier is trained and evaluated on the Isolated Gesture Recognition data set from the 2016 International Conference on Pattern Recognition (ICPR) Challenge. The data set comprises 47,933 gesture videos represented in color (RGB) and gray-scale depth Kinect (RGB-D) format.[**?** ] Each video depicts an isolated gesture, such as "peace sign" or "waving". 249 total distinct gestures are performed across 21 individuals. The data is preemptively split into train, validation, and test sets.

All in the dataset are downsampled by sampling frames at a fixed interval of 10 fps using FFmpeg [8], at a fixed. This produces, on average, 40-100 frames per video. Each `(C, H, W)`-dimensional frame for a given video is stacked along a time-step dimension to produce a four-dimensional `(T, C, H, W)` tensor, where `C` represents the number of channels, and `H` and `W` the height and width (240 and 360, respectively). The inputs are normalized with zero mean and unit variance across each of the three channels, and then a $224 \times 224$ center crop of the image is extracted.

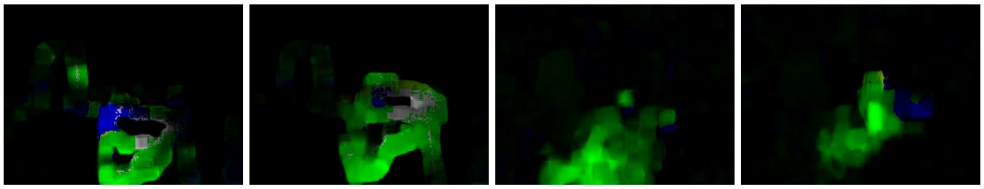

Figure 2: The results of optical flow analysis on the video frames illustrated in Figure 1.

# 4 Methods

## 4.1 Cross-Entropy Loss

Both our ResNet and LSTM models employ cross-entropy loss, the standard loss function for softmax classification. Cross-entropy measures the difference between a true probability distribution $p$ and estimated distribution $q$ as $H(p, q) = -\sum_x p(x) \log q(x)$. This value is minimized when the probability distribution of $q$ closely matches that of the true distribution $p$. For single-class classification, the target probability distribution is represented by a one-hot vector, where the value at the target index absorbs all the probability mass of 1. Thus, the cross-entropy loss for a single example is given by the negative log-likelihood if the softmax probability value given to the correct class label.

## 4.2 L2 Regularization

Regularization is a technique used to prevent the model from overfitting the training data. L2 regularization involves adding the L2-norm of a model's learned parameters to its loss function. When the L2-norm is large, the model's parameters are likely too complex and thus overfitting the data. This occurs when a disproportionate amount of weight is placed on a small subset of the features. Though this may yield high training accuracy, these weights are usually sparse, and as a result perform poorly when used to evaluate unseen examples that may not exactly resemble the training set. As a result, the model is conservative about increasing the value of its weights, and thus distributes weight more uniformly among the features. A well-regularized model is able to generalize to the validation and test set.

## 4.3 Residual Networks

Residual networks [9] are a particular class of convolutional neural networks (CNNs) that manage to learn well even when the network is structured in a deep configuration. In particular, residual networks combine the input of the residual network with the output. Typically, the output of a neural network is a function of some affine transformation $F(x) = Wx + b$. For a residual network, the output is given as $F(x) + x$. It is thus easier for the function to learn the identify mapping (when $F(x) = 0$), so during backpropagation, the vanishing gradient issue is dodged, since the network is more likely to resemble the identity function by default.

## 4.4 Lucas-Kanade Optical Flow

Lucas-Kanade Optical Flow [7] is a technique for optical flow estimation, which measures the apparent motion of objects caused by movement of the observer. It operates on two assumptions:

- Optical flow is nearly constant within the neighborhood of each pixel.
- Pixel intensities of an object do not change across frames.

Our project utilizes `OpenCV`'s [10] implementation of the algorithm, `calcOpticalFlowFarneback` [11], also known as "dense" optical flow. It returns two separate matrices, representing horizontal and vertical displacement for each pixel in the frame. These two displacement vectors are converted into HSV (hue, saturation, value) color space and stored as an image.

### 4.5 Model Architecture

The preprocessing step involves training 3 separate deep residual networks, mirroring the structure of the 18-layer residual network (`ResNet18`) exported by `pytorch.torchvision.models` [9]. The three networks are used to in order to individually encodes RGB-D frames, optical flow frames of RGB videos, and optical flow frames of RGB-D videos. For RGB videos, default pretrained weights (trained on the ImageNet challenge) are used for encoding.

The residual networks are trained as part of a supervised classification task, wherein individual frames of each video are treated as independent inputs, and the target label is equal to that of the frame's corresponding video. The residual network processes each frame, producing a 1000-dimensional vector. It passes this vector through a fully-connected layer with softmax output representing the output classification score. Thus, the residual network learns an encoding for each frame – one that picks out gesture-discriminative features. Training is expensive due to the sheer number of videos in our data set and the depth of the network, so this is trained once for each input type (attaining roughly 50% train accuracy and 2-3% validation accuracy), and then used to encode all frames in the dataset.

The LSTM network processes each input video, which at this point is represented as sequence of $T\times$ 1000-dimensional frame encodings, frame-by-frame. The final output of the LSTM, another 1000-dimensional vector, is fed through 3 fully-connected layers with ReLU activation and mapped to an output space with scores representing the 249 classes.

## 5 Results

### 5.1 Hyperparameters

Training is performed on a fixed minibatch size of 50, a choice that is based on CPU/GPU constraints and the number of videos in the dataset. Our experiments perform hyperparameter search over the learning rate, dropout, and weight decay (L2 regularization).

### 5.2 Evaluation

Our classifier is evaluated in terms of top prediction accuracy. Each prediction is simply the argmax of the network's output; our evaluation metrics only consider the top accuracy, although it could be useful to know whether the true class label lies in the top-$k$ classes ranked by prediction score. The accuracy of the model can be compared with those from the published results of many other models that competed in the ICPR '16 challenge. For this challenge, the state-of-the-art accuracy was 56.9% [5].

### 5.3 Experimental Results

The table below represents result of random hyperparameter sweeping on the `RGBD-OpticalFlowRGBD` LSTM model.

| Hyperparameters | Train | Valid | Test |
|---|---|---|---|
| lr=0.007, w=0.0008, d=0.07 | 2.37% | 2.16% | 2.30% |
| lr=0.008, w=0.0003, d=0.11 | 2.37% | 2.16% | 2.30% |
| lr=0.01, w=0.00, d=0.10 | 2.37% | 2.16% | 2.30% |

For an unknown reason, our model achieved identical performance across all swept hyperparameters over a small number of epochs. The accuracy quickly reaches a plateau at 2.30% test accuracy and 2.16% validation accuracy, as seen in Figure 3.

#### 5.3.1 Baseline Model Comparisons

The highest classification accuracy across each combination of features (RGB, RGBD, OFRGB (optical flow RGB), and OFRGBD (optical flow RGBD) are presented in the table below. Note that the LSTM model is identical across each of these experiments, so only the input features are compared.


Figure 3: **(Left) RGBD-OFRGBD LSTM Classifier.** Training vs. Validation Accuracy for the RGBD-OFRGBD LSTM classifier. **(Right) RGB-RGBD LSTM Classifier.** Training vs. Validation Accuracy for the baseline RGBD-RGBD LSTM classifier.

| Features | Train | Valid | Test |
|---|---|---|---|
| RGB | 78.71% | 6.09% | 8.13% |
| RGB-RGBD | 65.21% | 7.00% | 7.24% |
| RGBD-OFRGBD | 2.37% | 2.16% | 2.30% |

## 6 Conclusion/Future Work

The inclusion of optical flow features as input to our model did not improve classification accuracy of the model. The highest performing model was achieved by the RGB LSTM model, which does not include optical flow features. The test performance of the RGB and RGB-RGBD model do not differ by much, though it is likely that the performance is sensitive to hyperparameter choices, and not enough values were swept.

Lucas-Kanade optical flow justifiably produces poor feature encodings on RGB-D images, since these images depict grayscale objects moving through space. RGB-D videos violate both the pixel intensity and neighborhood uniformity assumptions posed by Lucas-Kanade optical flow estimations. Depth images are likely to involve varying pixel intensity across adjacent frames, since the value of a pixel measures an object's distance relative to the camera. If a hand moves toward the camera, then the pixel intensity will consequently change; as a result, Lucas-Kanade will not attribute any motion to the hand. This is a likely explanation for why the `RGBD-OpticalFlowRGBD` LSTM model yields low classification accuracy.

I suspect that there exists a bug in the implementation of the RGBD-OFRGBD LSTM model, because the training accuracy and loss did not change after several iterations. Theoretically, the model should be able to overfit to the training set as it did for the RGB and RGB-RGBD LSTM models. If given more time and computational resources, I would further debug and investigate these issues. I would also evaluate the LSTM on RGB-OFRGB inputs, because this is most similar to the two-stream classifier [6] which performed well on a separate dataset. Unfortunately, the final implementation failed to deserialize OFRGBD frames, possibly due to a GPU/software compatibility issue, and time/money restrictions precluded us from remedying these issues, so these results could not be gathered. They would have been very significant to the results and conclusions of this project.

Future directions for this project involve running the model for longer periods of time and experimenting with denser frame sampling (>10 fps), since this is a hyperparameter to which optical flow may be particularly sensitive. In addition, it would be worthwhile to replicate the exact two-stream convolutional network [6] network on the ICPR '16 challenge. Outside of using optical flow, I would recommend training separate models for hand, face, and body detection to improve performance. Many of the gestures are similar to one another, so giving special treatment to these bodily components are likely to improve classification accuracy.

Furthermore, there is a noisy degree of variation in the videos; actors performing the same gesture are seen performing gestures at different distances or in different quadrants of the video frame. In addition, some actors had different interpretations for how exactly to sign a gesture. Perhaps the dataset can be cleaned, or additional preprocessing steps can be used to normalize the position and size of each actor in the video.

## 7 Contributions

All portions of the project concerning optical flow features were carried out by the author, Kenny Leung, and serve as the main difference between the CS230 and CS231N submissions. Many thanks to Lucio Dery for his advice and support throughout the development of this project.

The code was jointly implemented by Kenny Leung, Ying Hang Seah, and Hiroshi Mendoza. All members were involved in experimentation. A breakdown of contributions is as follows:

- Kenny: Experiment configuration and logging schema, hyperparameter sweeper, basic data loaders, training and validation logic, visualizations and analysis, baseline LSTM classifier with a pretrained `ResNet18` CNN model, all optical flow related models.
- Ying Hang: Improvements to configuration and logging, CUDA integration, combination data loader, testing logic, custom `ResNet18` LSTM models, feature normalization and custom `ResNet18` models for RGB-D inputs.
- Hiroshi: Dataset acquisition and cleaning, frame sampling, GPU handling, explored hardware integration to connect the classifier with an Intel Realsense depth camera.

## References

[1] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[2] P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, "Large-scale isolated gesture recognition using convolutional neural networks," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pp. 7–12, IEEE, 2016.

[3] O. Özdemir, N. C. Camgöz, and L. Akarun, "Isolated sign language recognition using improved dense trajectories," in *Signal Processing and Communication Application Conference (SIU), 2016 24th*, pp. 1961–1964, IEEE, 2016.

[4] Y. Li, K. Tian, Y. Fan, X. Xu, and R. Li, "Video gesture recognition with rgb-ds data based on 3d convolutional networks," in *ICPR*, 2016.

[5] P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, "Large-scale isolated gesture recognition using convolutional neural networks," *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016.

[6] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, pp. 568–576, 2014.

[7] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[8] F. Bellard, M. Niedermayer, *et al.*, "Ffmpeg," *Available from: http://ffmpeg. org*, vol. 3, 2012.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, 2000.

[11] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*, pp. 363–370, Springer, 2003.