Final Project: Human Action Recognition
Hayleigh Sanders, RIN# 661195735

Introduction

The following project is an implementation of the two-stream CNN model proposed by Simonyan and Zisserman [1] for action recognition in videos using the UCF11 Youtube action dataset. In this model, the separate CNN outputs of an image sequence and its corresponding optical flow sequence are averaged before softmax is applied to the result so that spatial and temporal information can be factored into the action class selection.

Theory and Setups

Action recognition from a series of video frames can be effectively computed from the synthesis of spatial and temporal data through the use of a two-stream CNN, shown below.

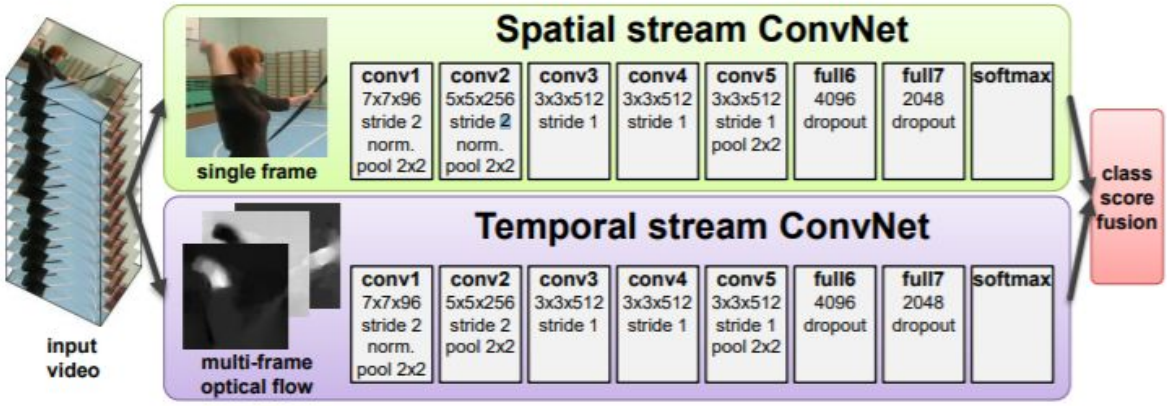

Figure 1: Two stream CNN model architecture, image credit Simonyan et al [1]

Spatial CNN: Image features

The first CNN stream contains information about the spatial properties of the image sequence. Images in a sequence are fed through a CNN with several layers to learn the static features associated with each action class, ie objects or settings such as animals, balls or athletic courts. The output is a 1x11 vector corresponding to the class probability predictions for each image.

Temporal CNN: Optical flow

The second CNN stream contains information about the temporal properties of the image sequence using the optical flow between images as an input. Its output is a 1x11 vector corresponding to the class probability predictions. Optical flow is best understood as the displacement of brightness patches between two images relative to an observer corresponding to pixel motion, and can be used to encode temporal information between images in a

sequence. The Horn-Schunck method was employed to compute the optical flow for a given image frame based on its next frame.

Suppose an image p=(x,y) has an underlying flow field w(p) = (u(p),v(p), 1), where u(p) and v(p) are the horizontal and vertical components of the flow field. Under the brightness constancy assumption,

$$I(x(t), y(t), t) = c \quad (1)$$

The brightness of a point across a sequence of images is assumed to be constant. Thus the objective function can be formed,

$$E(u, v) = \int \left| I_2(p + w) - I_1(p) \right|^2 + \lambda\left(|\nabla u|^2 + |\nabla v|^2 \right) dp \quad (2)$$

Where I_1 and I_2 are two images in a sequence and λ is the regularization weight. This equation can be solved with an iterative flow framework that assumes w is an estimate of the flow field that can be updated incrementally with dw = (du,dv). Therefore, equation (1) can be substituted with,

$$E(du, dv) = \int \left| I_2(p + w + dw) - I_1(p) \right|^2 + \lambda\left(|\nabla (u + du)|^2 + |\nabla (v + dv)|^2 \right) dp \quad (3)$$

If the following is defined as,

$$\begin{aligned} I_z(p) &= I_2(p + w) - I_1(p) \\ I_x(p) &= \frac{\partial}{\partial x} I_2(p + w) \\ I_y(p) &= \frac{\partial}{\partial y} I_2(p + w) \end{aligned} \quad (4)$$

The first order Taylor expansion will be,

$$I_2(p + w + dw) - I_1(p) \approx I_2(p) + I_x(p)\, du(p) + I_y(p)\, dv(p) \quad (5)$$

An example output of optical flow using the Horn-Schunck method is shown below for two consecutive image frames from a sequence in the training data set. The optical flow in both the x and y directions is plotted on the second image with a quiver map. While the human eye may not be able to detect much difference between frames, the algorithm was able to pick up the movement of the person and basketball distinctly from a static background.

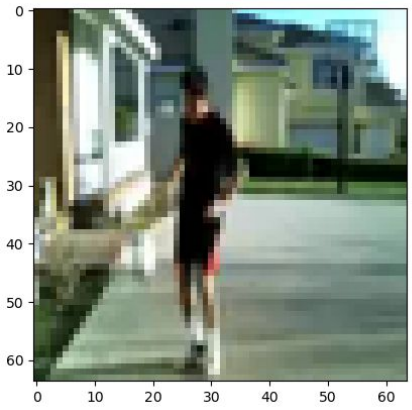

Figure 1: Frame at t-1

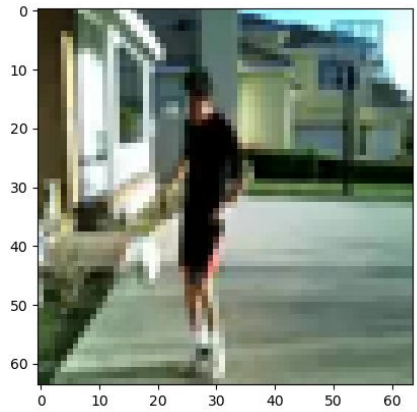

Figure 2: Frame at t

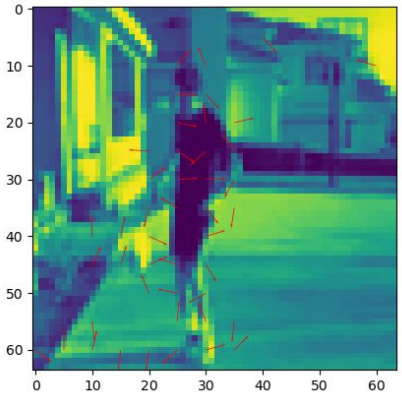

Figure 3: Optical flow between frames, plotted on Figure 2

Model Structure and Implementation

The dataset consists of 7272 sequences of 30 RGB 64x64 pixel images from the UCF11 Youtube action database. Each sequence is associated with one of 11 action classes: "basketball shooting", "biking/cycling", "diving", "golf swinging", "horse back riding", "soccer juggling", "swinging", "tennis swinging", "trampoline jumping", "volleyball spiking" and "walking with a dog". A subset of 5000 images was chosen from the dataset to serve as the training dataset, and the remaining 2272 were used as the testing dataset to validate the model. The data was normalized before being processed by the CNNs by finding the maximum and minimum values in each image, then converting each pixel x to (x - minimum value)/(maximum value - minimum value).

The training dataset was shuffled randomly and batches of five image sequences were selected for each training iteration. These sequences were then fed into a CNN. The CNN used in this implementation consisted of one convolutional layer with a [3,3,3,32] filter, a ReLu activation layer and one max pooling layer. The second convolutional layer used a [3,3,32,32] filter, a Relu activation layer and a max pooling layer. The third convolutional layer used a [3,3,32,64] filter

and a ReLu activation layer. Next, the result was flattened and fed to three consecutive fully connected layers of sizes 128, 256 and 512 respectively. The final output contains a [1x11] array of class predictions for each image.

The spatial result is stored in a variable and the image sequence is then converted to a sequence of optical flow images using the Horn-Schunck algorithm using equations 1 through 5. Each image needed to be converted to grayscale before the algorithm could be applied, and each result was a [64,64,2] tensor representing the x,y components of optical flow for each of the pixels in the image. This sequence of optical flow tensors was then fed into a CNN with the same structure as the previous spatial data step. The result corresponds to a [1x11] list of class predictions for each image. These prediction vectors are then averaged with the prediction vectors from the spatial data. This combined result was then used to compute the loss, which was fed into the optimizer to train the model. AdamOptimizer was used for this implementation. The loss and accuracy for the model was computed for the training data batch and the testing data at each epoch. This combined model will theoretically factor both spatial and temporal information to classify each sequence as an action.

The loss function used was the tf.nn.softmax_cross_entropy_with_logits function inside the tf.reduce_mean function, which computes the softmax cross entropy between the combined CNN outputs and the class labels, which were one-hot encoded into [1x11] vectors equal to one at the index of the correct class labels and zero otherwise. Softmax is given by,

$$\sigma_M(z) = \frac{exp(z)}{\sum_{k'=1}^{K} exp(z)} \quad (6)$$

Cross-entropy is also given by,

$$L(y,\hat{y}) = -\frac{1}{M}\sum_{i=m}^{M} (y^m \log(\hat{y}^m) + (1-y^m)\log(1-\hat{y}^m)) \quad (7)$$

This output function produces a probability distribution corresponding to the class predictions for each of the 11 classes. This loss function was used because the model must perform a multiclass classification task.

Hyperparameters

The learning rate used with AdamOptimizer was set to .01 by trial and error through observing the rate at which the model improved its class predictions and adjusting accordingly. The batch size was set to 5 because a relatively large amount of data was being processed at once for each training iteration and a large batch size would slow down the training process by a considerable degree. 100 epochs were chosen as the number of training iterations by observing

the point at which the change in loss began to plateau. 5000 sequences were chosen as the training data size so that enough data would be left over to compute relatively accurate testing loss and error statistics. The filter sizes for the convolutional neural network were set to [3,3,3,32], [3,3,32,32] and [3,3,32,64] based on the input image size. No padding was used for each convolutional layer, and the strides were either [1,1,1,1] or [1,2,2,1] because the input images were relatively small. ReLu was used as the activation function between convolution layers to add nonlinearity.

Experimental Results

The training and testing classification accuracy vs the training epochs is plotted below.

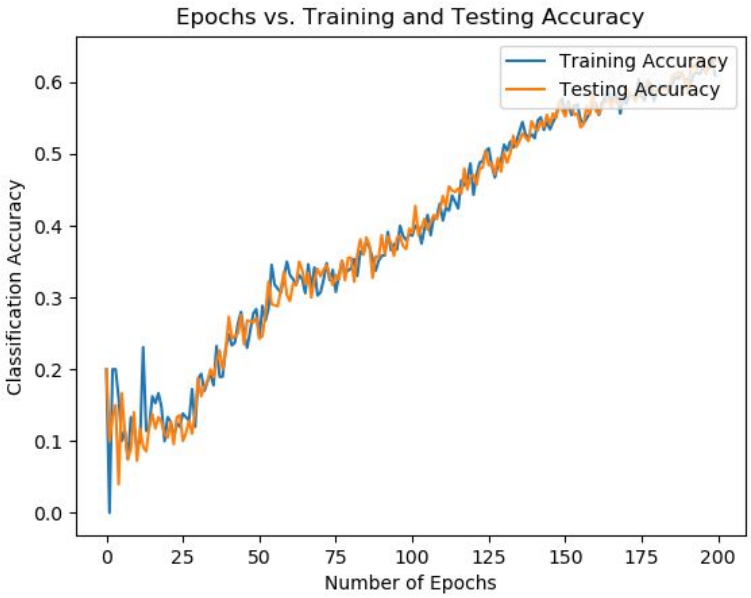

Figure 4: Training and testing accuracy vs iterations

The loss for the training and testing datasets vs the epochs is plotted below.

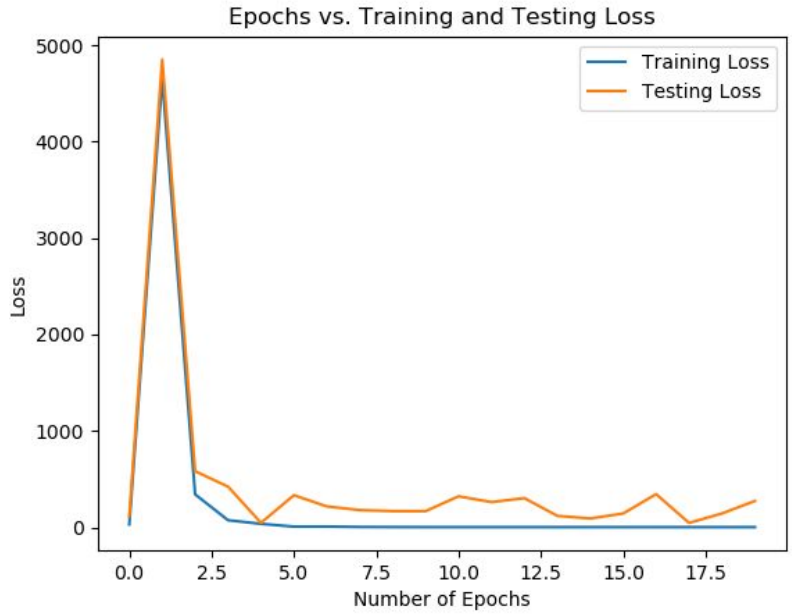

Figure 5: Loss vs Epochs for training and testing data

The classification error of the final model was computed for the testing data and is plotted below. Each numbered class is equal to 0: "basketball shooting", 1: "biking/cycling", 2: "diving", 3: "golf swinging", 4: "horse back riding", 5: "soccer juggling", 6: "swinging", 7: "tennis swinging", 8: "trampoline jumping", 9: "volleyball spiking" and 10: "walking with a dog".

Figure 6: Classification error by class

The confusion matrix is plotted below.

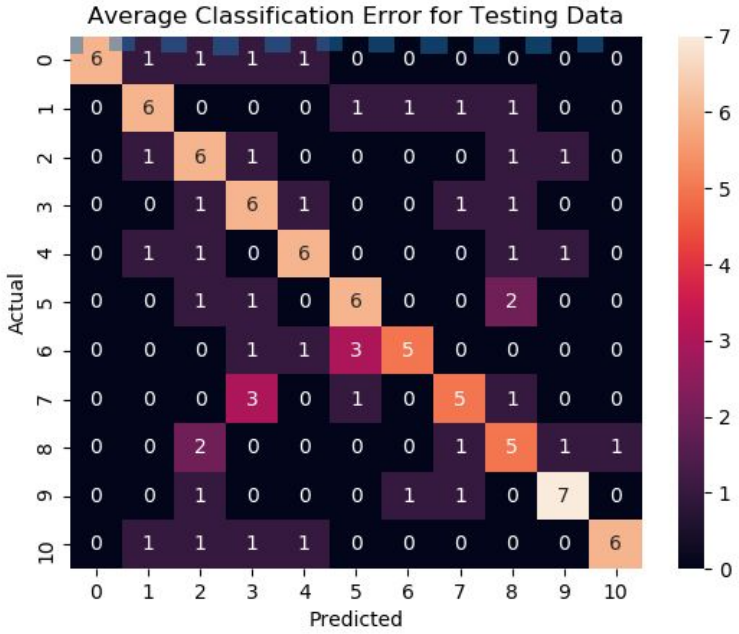

Figure 7: Confusion matrix

Conclusion

The accuracy approached 60% over a large number of iterations, but overall this implementation was not as accurate as the model referenced in the paper. This may have to do with the smoothness assumption made by the Horn-Schunck algorithm in estimating optic flow. Noisy data may have interfered with the ability of the algorithm to compute optic flow properly.

Reference

[1] Simonyan, Karen, and Andrew Zisserman. ”Two-stream convolutional networks for action recognition in videos.” In Advances in neural information processing sys- tems, pp. 568-576. 2014.