Lab 5: Implementing a Convolutional Neural Network with Keras for action recognition in videos

October 31, 2019

1 Introduction

In this lab, we have built a convolutional neural network with the Keras framework in Python aimed at recognizing actions in videos. We havel use the UCF101 dataset, that gathers Youtube videos in 101 categories. Some actions to recognize are: "Apply eye makeup", "Apply lipstick", "Playing guitar", "Playing violin", "Skate boarding", "Skiing", ... In order to classify actions, we have used a Siamese Convolutional Neural Network. This network will take both a slice of the video and the corresponding optical flow, and produce the probabilities to belong to the 101 action classes. We will build this network progressively, one branch at a time, and study the contribution of each architectural change.
Each branch has the same architecture:
Conv 3D→ MaxPooling → Conv 3D → MaxPooling → Conv 3D → MaxPooling → F.C. → F.C.

The Conv. Layers all have filters of size 3, zero padding and Relu as activation function. The number of filters is respectively 30, 60 and 80. The MaxPooling layers all have a size of 2. For the first Fully Connected layer we will use 500 neurons.

2 Task 1: Convolutional Neural Network for action recognition: Color Branch

In this task we have programmed the first convolutional neural network to recognize the 101 actions using color information from videos which is basically similar to one branch of Siamese Network. To do this, we have developed a python function named *make_one_branch_model()* to define the model of color branch. The architecture of the network is given in figure (1) which we will use in second task as well to train the network. We have tested the network for both RGB and Flow data after loading pretrained weights. The Accuracy of the network is given in Table (1) .

Table 1: Accuracy of the Network for both RGB and Flow Data

	Test Data Accuracy	Test Data Loss
with RGB Data	43.33%	2.32419
with Flow data	23.33%	2.95846

3 Task 2: Convolutional Neural Network for action recognition: optical flow branch

In this task we have trained the convolutional neural network to recognize the 101 actions using optical flow from videos. To do this, we have used the previously developed python function *make_one_branch_model()*. The architecture of the network is given in figure 1 which is same as the Task 1 Network. In this task we have trained the network for 30 epochs and evaluated the performance. The Accuracy of the network is given in Table (2) .

In Table (2) All the training have been done with ***Batch Size=10***.

Table 2: Accuracy of the Network for both Flow and RGB data after Re-Training the model for 30 epochs

	Training Accuracy	Training Loss	Test Accuracy	Test Loss
with Flow Data	96.67%	0.1693	0.0525	100.00%
with RGB data	100.00%	0.0060	0.0581	100.00%

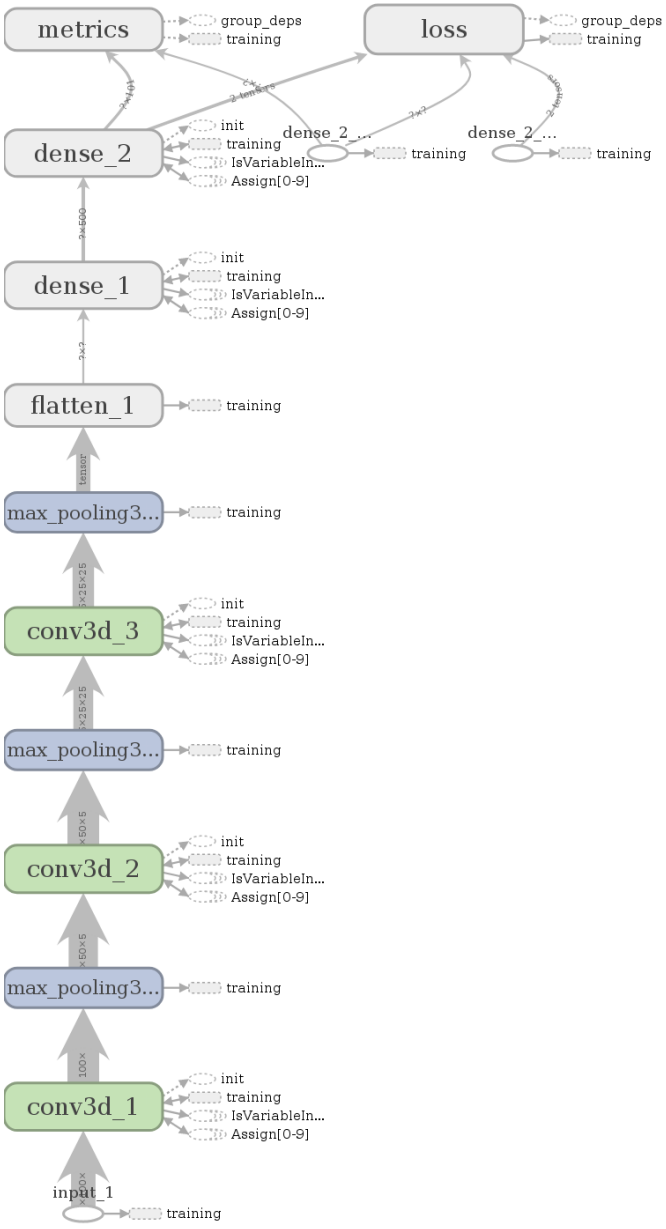
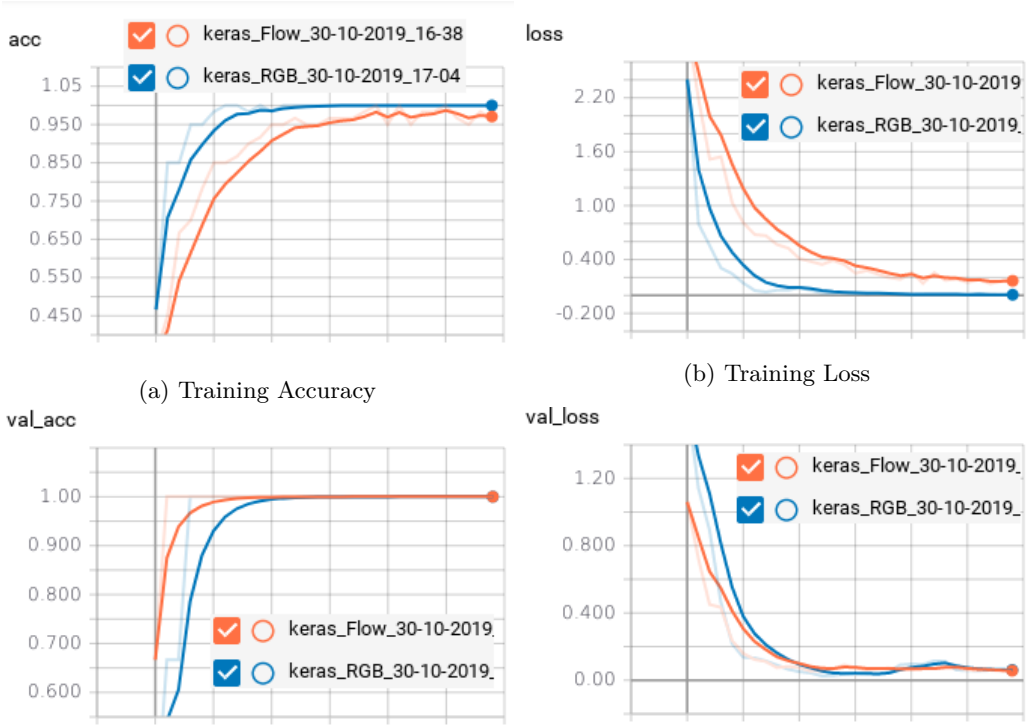


Figure 1: One Branch of the Siamese Network

In figure (2), we can see the training progress of the network for 30 epochs with Batch Size=10. Orange line represents flow data training and blue-ish line represents RGB data training. In training we are using testing data for validation purpose as well. For both seperately trained RGB and Flow data cases, we have achieved 100% accuracy on Testing data.



(a) Training Accuracy

(b) Training Loss

(c) Validation Accuracy

(d) Validation Loss

Figure 2: Training of One Branch of Siamese Network for Flow and RGB data seperately for 30 epochs

4 Task 3: Convolutional Neural Network for action recognition: siamese network

In this task, we have developed a convolutional neural network to recognize the 101 actions using both color information and optical flow from videos. This Siamese network will merge the output of the two branches [before their last F.C. layer] with a concatenation layer. The network architecture is given in figure (3). To concatenate, we have used *keras.layers.concatenate()*. In this task, after defining the Siamese model by the python function *make_model()*, we loaded the pretrained weights and later we have retrained the network for 30 epochs. Unlike previous task, we could not use Batch size 10 as we run out of GPU memory. So we have retrained the network with Batch Size 1, 3 and 5. The Accuracy of the Siamese network is given in Table (3).

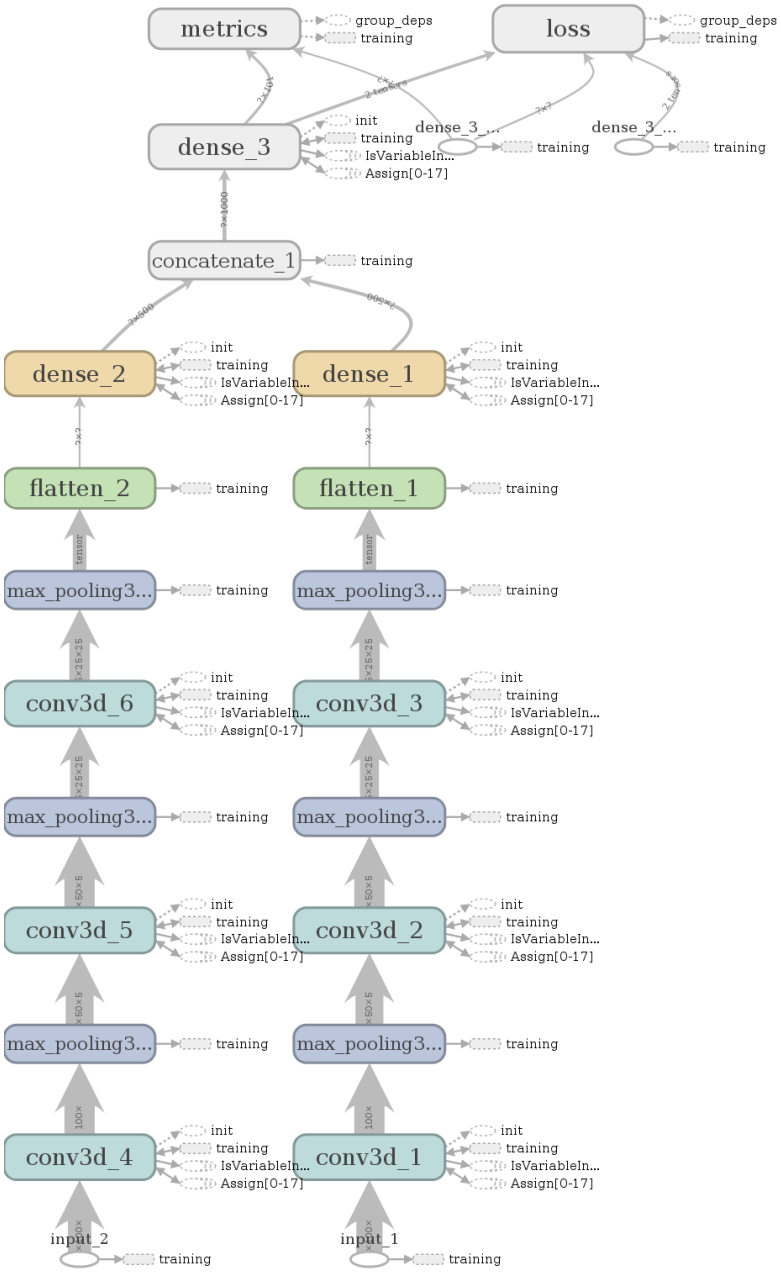


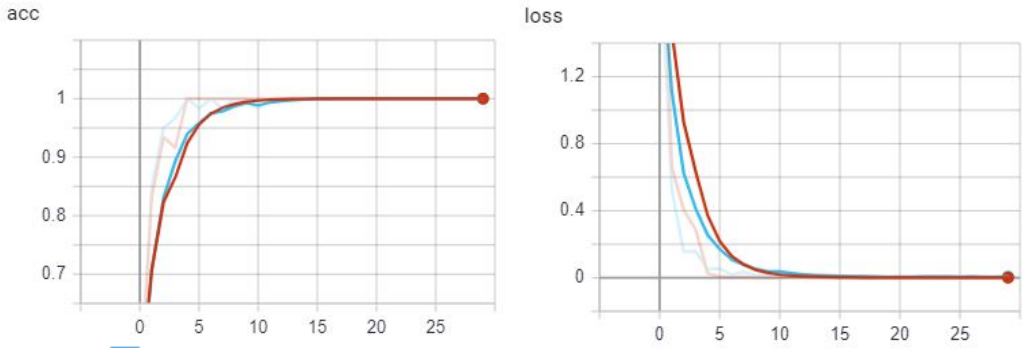
Figure 3: Architecture of Siamese Network

From the table (3), we can see that if we decrease batch size, accuracy of the networks decreases.

Table 3: Accuracy of the Siamese Network for different Batch Size, Retrained for 30 epochs

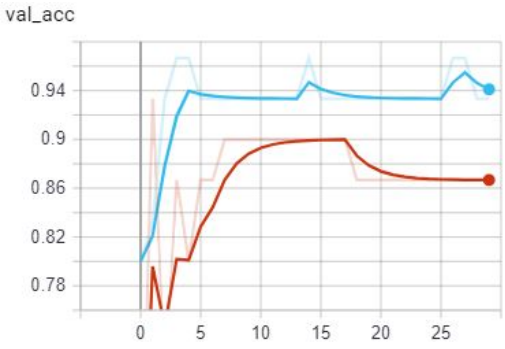
	Training Accuracy	Training Loss	Test Accuracy	Test Loss
Batch Size=1	100.00%	8.5843e-04	86.67%	0.4397
Batch Size=3	100.00%	0.0028	86.67%	0.3201
Batch Size=5	100.00%	0.0076	93.33%	0.2154

In figure (4), training progress for 30 epochs are given for Batch Size 1 and 5. Here, Reddish line represents Batch size 1 and Sky Blue Line represents Batch Size 5. It is visible that, the training process has achieved saturation within 30 epochs so if we train for more epochs then we may overfit the network. But if we could increase the Batch Size we may improve the accuracy of the network.

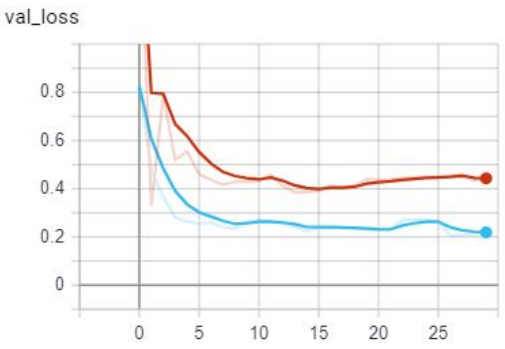


(a) Training Accuracy

(b) Training Loss



(c) Validation Accuracy



(d) Validation Loss

Figure 4: Training of Siamese Network for Batch Size 1(Reddish Line) and 5(Sky Blue Line) for 30 epochs