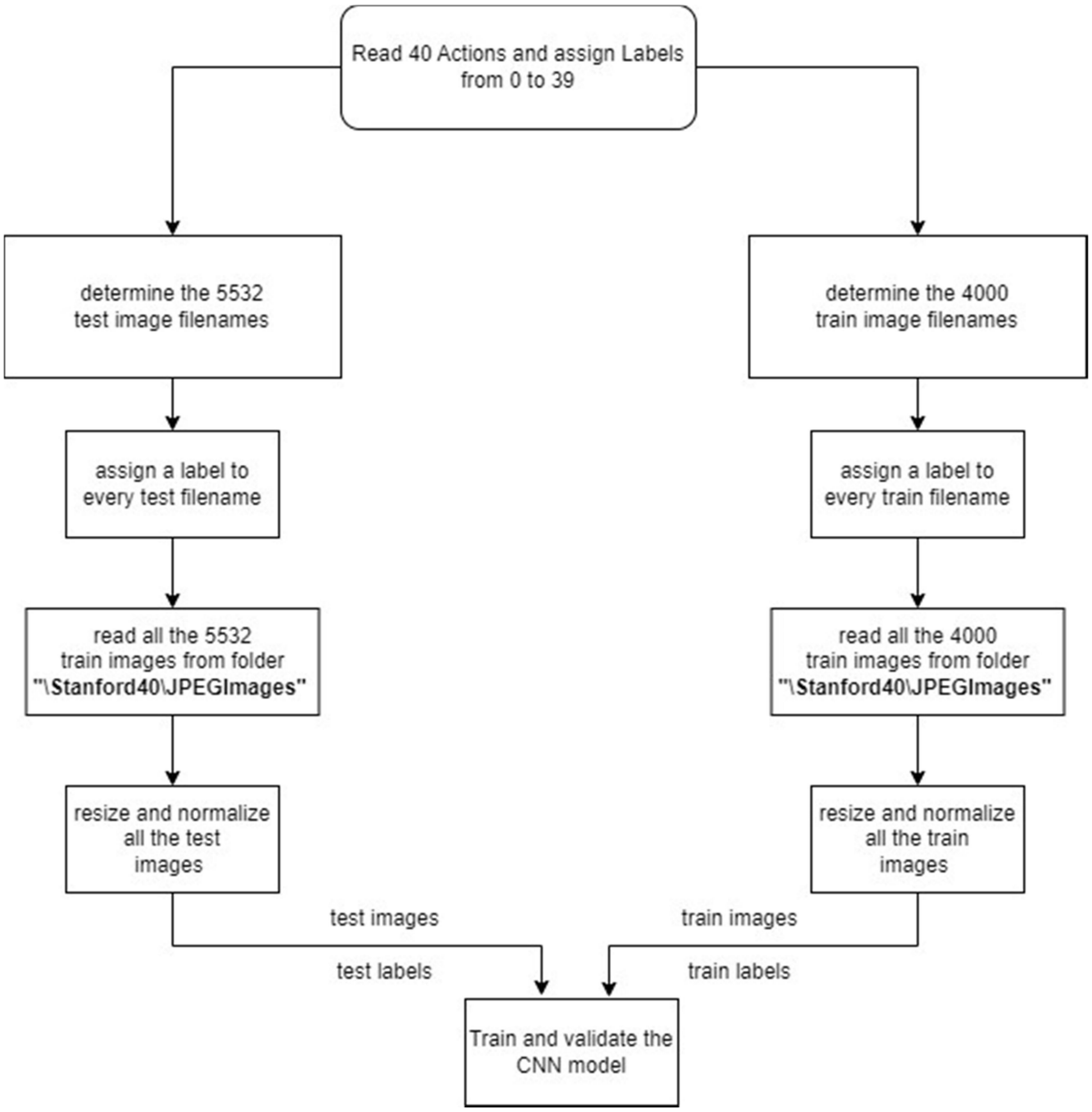
Assignment #2

Action Recognition From Still Images Using

Deep Learning Networks

- **Libraries used**
    1. OpenCV Python Library
    2. NumPy Python Library
    3. MatplotLib Python plotting Library
    4. Keras API in tensorflow platform
    5. Pandas tool
- **Algorithm**
    1. Read **actions.txt** file to determine the 40 actions names
    2. Assign labels from 0 to 39 to these actions in a dictionary type e.g. action **applauding** is assigned label **0**
    3. read and parse the **train.txt** file to determine the 4000 train image filenames
    4. assign a label to every train filename according to the name of the file e.g. file **applauding_001.jpg** is assigned **0** because it's name contains string "**applauding"** which is the action name
    5. read all the 4000 train images from folder **"\Stanford40\JPEGImages"** based on train image filenames determined above
    6. resize all the train images to be of size **128 * 128 * 3**
    7. normalize the pixel values of train images to be from -0.5 to 0.5 instead of 0 to 255
    8. repeat above steps 3, 4, 5, 6, 7 to obtain 5532 test images
    9. train and validate the CNN using Keras model with parameters shown in code ,CNN code similar to https://victorzhou.com/blog/keras-cnn-tutorial/ with parameters modified.



- **Results**

Run **Assignment2.ipynb** Jupyter notebook file to get the following results

The CNN was trained with 10 epochs the accuracy reached 99% at the last epoch as shown below starting from 0.0518 at the first epoch and the loss reached 0.1498 in the last epoch starting from 3.6484 in the first epoch

```
Epoch 1/10
125/125 [==============================] - 495s 3s/step - loss: 3.6484 - accuracy: 0.0518 - val_loss: 3.5600 -
val_accuracy: 0.0642
Epoch 2/10
125/125 [==============================] - 107s 853ms/step - loss: 3.2271 - accuracy: 0.1873 - val_loss: 3.5749 -
val_accuracy: 0.0638
Epoch 3/10
125/125 [==============================] - 29s 236ms/step - loss: 2.8308 - accuracy: 0.3575 - val_loss: 3.5871 -
val_accuracy: 0.0761
Epoch 4/10
125/125 [==============================] - 29s 232ms/step - loss: 2.3199 - accuracy: 0.5540 - val_loss: 3.6382 -
val_accuracy: 0.0757
Epoch 5/10
125/125 [==============================] - 29s 234ms/step - loss: 1.7182 - accuracy: 0.7383 - val_loss: 3.7690 -
val_accuracy: 0.0649
Epoch 6/10
125/125 [==============================] - 28s 229ms/step - loss: 1.1439 - accuracy: 0.8683 - val_loss: 3.9265 -
val_accuracy: 0.0649
Epoch 7/10
125/125 [==============================] - 29s 236ms/step - loss: 0.7067 - accuracy: 0.9440 - val_loss: 4.0995 -
val_accuracy: 0.0607
Epoch 8/10
125/125 [==============================] - 27s 215ms/step - loss: 0.4162 - accuracy: 0.9815 - val_loss: 4.2755 -
val_accuracy: 0.0573
Epoch 9/10
125/125 [==============================] - 25s 199ms/step - loss: 0.2441 - accuracy: 0.9930 - val_loss: 4.4495 -
val_accuracy: 0.0589
Epoch 10/10
125/125 [==============================] - 29s 230ms/step - loss: 0.1498 - accuracy: 0.9983 - val_loss: 4.5986 -
val_accuracy: 0.0591
```

- **Bibliograpgy**
  - https://victorzhou.com/blog/keras-cnn-tutorial/
  - https://victorzhou.com/blog/intro-to-cnns-part-1/
  - https://machinelearningmastery.com/building-a-convolutional-neural-network-in-pytorch/
  - http://vision.stanford.edu/Datasets/40actions.html