In [1]:

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="{}".format(2) # gpu idx
```

In [2]:

```
import warnings
warnings.filterwarnings(action='ignore')
import tensorflow as tf
from keras.models import Sequential, Model
from keras.layers import ZeroPadding3D, Input, Dense, Flatten, BatchNormalization,MaxPool3D, Activa
from keras.layers.convolutional import Conv3D
from keras import optimizers
from keras.optimizers import Adam, SGD
import keras
from keras import backend as K
import keras_video.utils
```

```
Using TensorFlow backend.
```

In [3]:

```
import keras_video
import glob
classes = [i.split(os.path.sep)[1] for i in glob.glob('train_five/*')]
glob_pattern='train_five/{classname}/*.webm'
```

In [4]:

```
train = keras_video.VideoFrameGenerator(classes=classes, glob_pattern=glob_pattern, nb_frames=16, ba
```

```
class attach, validation count: 100, train count: 400
class close, validation count: 100, train count: 400
class cover, validation count: 100, train count: 400
class drop, validation count: 100, train count: 400
class fold, validation count: 100, train count: 400
class hold, validation count: 100, train count: 400
class move, validation count: 100, train count: 400
class put, validation count: 100, train count: 400
class take, validation count: 100, train count: 400
class throw, validation count: 100, train count: 400
Total data: 10 classes for 4000 files for train
```

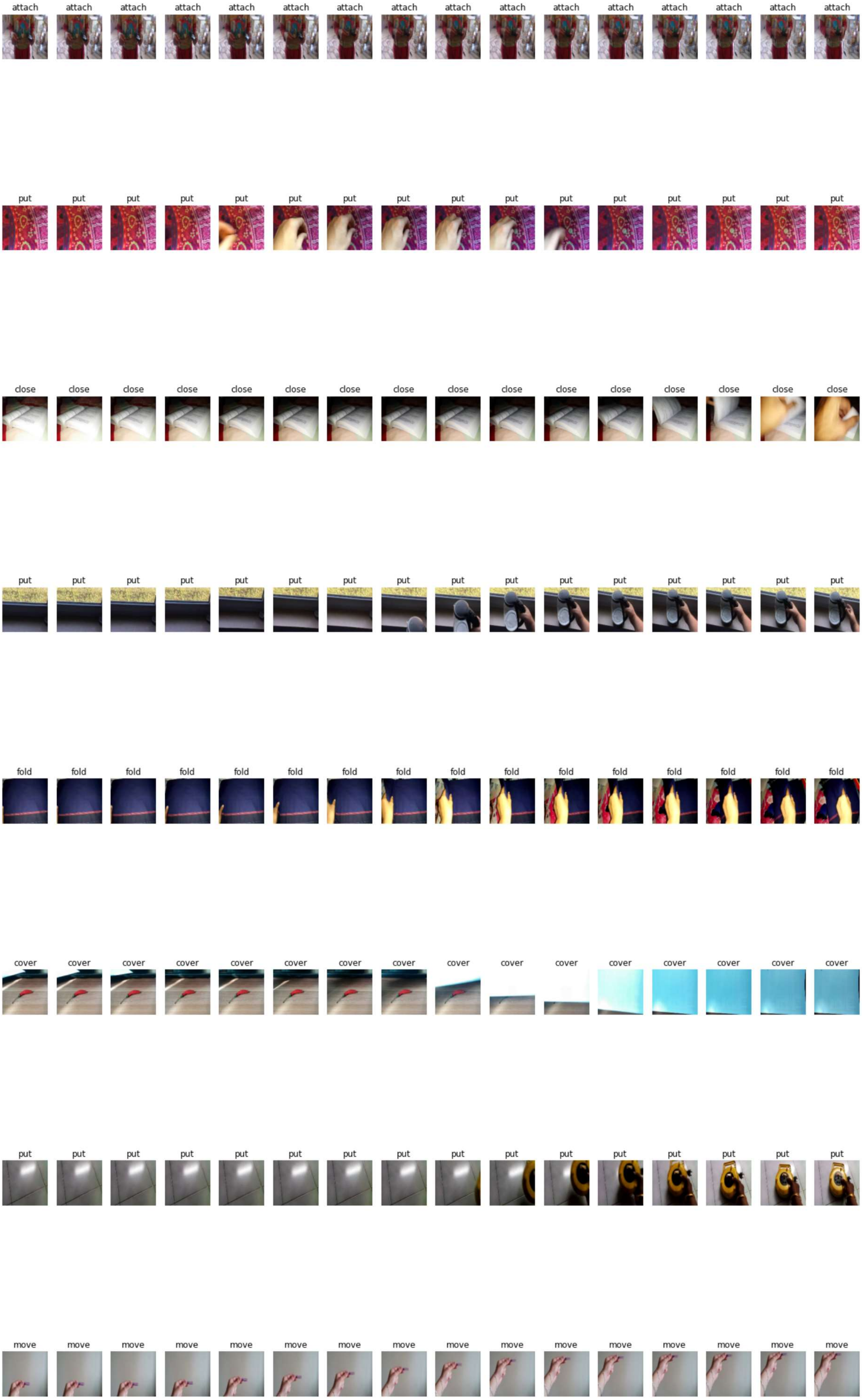
In [5]:

```
valid = train.get_validation_generator()
```

```
Total data: 10 classes for 1000 files for validation
```

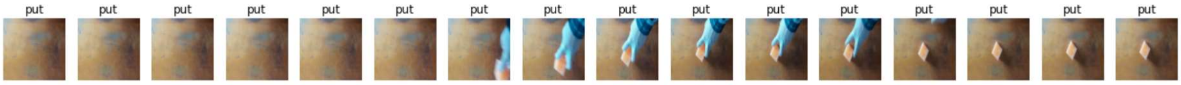
In [6]:

```
keras_video.utils.show_sample(train, random=True)
```


attach attach attach attach attach attach attach attach attach attach attach attach attach attach attach attach
put put put put put put put put put put put put put put put put
close close close close close close close close close close close close close close close close
put put put put put put put put put put put put put put put put
fold fold fold fold fold fold fold fold fold fold fold fold fold fold fold fold
cover cover cover cover cover cover cover cover cover cover cover cover cover cover cover cover
put put put put put put put put put put put put put put put put
move move move move move move move move move move move move move move move move

cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover

take
take
take
take
take
take
take
take
take
take
take
take
take
take
take
take


put
put
put
put
put
put
put
put
put
put
put
put
put
put
put
put

close
close
close
close
close
close
close
close
close
close
close
close
close
close
close
close

attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach

cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover
cover

take
take
take
take
take
take
take
take
take
take
take
take
take
take
take
take

take
take
take
take
take
take
take
take
take
take
take
take
take
take
take
take

throw
throw
throw
throw
throw
throw
throw
throw
throw
throw
throw
throw
throw
throw
throw
throw

attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach

attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach
attach

close
close
close
close
close
close
close
close
close
close
close
close
close
close
close
close

In [ ]:

```
## input layer
input_layer = Input((16, 120, 120, 3))

## convolutional layers
conv_layer1 = Conv3D(filters=8, kernel_size=(3, 3, 3), activation='relu')(input_layer)
conv_layer2 = Conv3D(filters=16, kernel_size=(3, 3, 3), activation='relu')(conv_layer1)

## add max pooling to obtain the most imformatic features
pooling_layer1 = MaxPool3D(pool_size=(2, 2, 2))(conv_layer2)

conv_layer3 = Conv3D(filters=32, kernel_size=(3, 3, 3), activation='relu')(pooling_layer1)
conv_layer4 = Conv3D(filters=64, kernel_size=(3, 3, 3), activation='relu')(conv_layer3)
pooling_layer2 = MaxPool3D(pool_size=(2, 2, 2))(conv_layer4)

## perform batch normalization on the convolution outputs before feeding it to MLP architecture
pooling_layer2 = BatchNormalization()(pooling_layer2)
flatten_layer = Flatten()(pooling_layer2)

## create an MLP architecture with dense layers : 4096 -> 512 -> 10
## add dropouts to avoid overfitting / perform regularization
dense_layer1 = Dense(units=2048, activation='relu')(flatten_layer)
dense_layer1 = Dropout(0.4)(dense_layer1)
dense_layer2 = Dense(units=512, activation='relu')(dense_layer1)
dense_layer2 = Dropout(0.4)(dense_layer2)
output_layer = Dense(units=10, activation='softmax')(dense_layer2)

## define the model with input layer and output layer
model = Model(inputs=input_layer, outputs=output_layer)
```

In [8]:

```
## input layer
input_layer = Input((16, 112, 112, 3))

## convolutional layers
conv_layer1 = Conv3D(filters=64, kernel_size=(3, 3, 3), activation='relu', padding='same', strides=
pooling_layer1 = MaxPool3D(pool_size=(1, 2, 2), strides=(1, 2, 2), padding='valid')(conv_layer1)

conv_layer2 = Conv3D(filters=128, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
pooling_layer2 = MaxPool3D(pool_size=(2, 2, 2), strides=(2, 2, 2), padding='valid')(conv_layer2)

## add max pooling to obtain the most imformatic features

conv_layer3 = Conv3D(filters=256, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
conv_layer4 = Conv3D(filters=256, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
pooling_layer3 = MaxPool3D(pool_size=(2, 2, 2), strides=(2, 2, 2), border_mode='valid')(conv_layer4

conv_layer5 = Conv3D(filters=512, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
conv_layer6 = Conv3D(filters=512, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
pooling_layer4 = MaxPool3D(pool_size=(2, 2, 2), strides=(2, 2, 2), padding='valid')(conv_layer6)
conv_layer7 = Conv3D(filters=512, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
conv_layer8 = Conv3D(filters=512, kernel_size=(3, 3, 3), activation='relu', padding='same', strides
pooling_layer5 = ZeroPadding3D(padding=(0, 1, 1))(conv_layer8)
## perform batch normalization on the convolution outputs before feeding it to MLP architecture
pooling_layer6 = MaxPool3D(pool_size=(2, 2, 2), strides=(2, 2, 2), padding='valid')(pooling_layer5)

flatten_layer = Flatten()(pooling_layer6)

## create an MLP architecture with dense layers : 4096 -> 512 -> 10
## add dropouts to avoid overfitting / perform regularization
dense_layer1 = Dense(units=4096, activation='relu')(flatten_layer)
dense_layer1 = Dropout(0.5)(dense_layer1)
dense_layer2 = Dense(units=4096, activation='relu')(dense_layer1)
dense_layer2 = Dropout(0.5)(dense_layer2)
output_layer = Dense(units=10, activation='softmax')(dense_layer2)

## define the model with input layer and output layer
model = Model(inputs=input_layer, outputs=output_layer)
```

In [9]:

```
optimizer = Adam(learning_rate=1e-5)
#optimizer = SGD(learning_rate=1e-6)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
print (model.summary())
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 16, 112, 112, 3)   0         
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 16, 112, 112, 64)  5248      
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 16, 56, 56, 64)    0         
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 16, 56, 56, 128)   221312    
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 8, 28, 28, 128)    0         
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 8, 28, 28, 256)    884992    
_________________________________________________________________
conv3d_4 (Conv3D)            (None, 8, 28, 28, 256)    1769728   
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 4, 14, 14, 256)    0         
_________________________________________________________________
conv3d_5 (Conv3D)            (None, 4, 14, 14, 512)    3539456   
_________________________________________________________________
conv3d_6 (Conv3D)            (None, 4, 14, 14, 512)    7078400   
_________________________________________________________________
max_pooling3d_4 (MaxPooling3 (None, 2, 7, 7, 512)      0         
_________________________________________________________________
conv3d_7 (Conv3D)            (None, 2, 7, 7, 512)      7078400   
_________________________________________________________________
conv3d_8 (Conv3D)            (None, 2, 7, 7, 512)      7078400   
_________________________________________________________________
zero_padding3d_1 (ZeroPaddin (None, 2, 9, 9, 512)      0         
_________________________________________________________________
max_pooling3d_5 (MaxPooling3 (None, 1, 4, 4, 512)      0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              33558528  
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312  
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                40970     
=================================================================
Total params: 78,036,746
Trainable params: 78,036,746
Non-trainable params: 0
_________________________________________________________________
None
```

In [10]:

```
EPOCHS=30

callbacks = [
    #keras.callbacks.ReduceLROnPlateau(verbose=1),#측정 항목이 향상되지 않는 경우 learning rate 줄임
    keras.callbacks.ModelCheckpoint(
        '/home/jiho/work/repos/somth2smth_conv3d/model/weights.{epoch:02d}-{val_loss:.2f}.hdf5',
        verbose=1),
    keras.callbacks.TensorBoard(log_dir='/home/jiho/work/repos/somth2smth_conv3d')#그래프로 보기위해
]
model.fit_generator(
    train,
    validation_data=valid,
    verbose=1,
    epochs=EPOCHS,
    callbacks=callbacks
)
```

```
200/200 [==============================] - 149s 743ms/step - loss: 1.1376 - accur
acy: 0.6143 - val_loss: 1.0841 - val_accuracy: 0.4880

Epoch 00022: saving model to /home/jiho/work/repos/somth2smth_conv3d/model/weight
s.22-1.08.hdf5
Epoch 23/30
200/200 [==============================] - 148s 740ms/step - loss: 1.0392 - accur
acy: 0.6435 - val_loss: 1.9013 - val_accuracy: 0.4930

Epoch 00023: saving model to /home/jiho/work/repos/somth2smth_conv3d/model/weight
s.23-1.90.hdf5
Epoch 24/30
200/200 [==============================] - 148s 741ms/step - loss: 0.9965 - accur
acy: 0.6597 - val_loss: 1.4739 - val_accuracy: 0.5090

Epoch 00024: saving model to /home/jiho/work/repos/somth2smth_conv3d/model/weight
s.24-1.47.hdf5
Epoch 25/30
200/200 [==============================] - 148s 742ms/step - loss: 0.9279 - accur
acy: 0.6845 - val_loss: 1.1828 - val_accuracy: 0.5050
```

In [ ]: