* Dùng SVM thuần cho bài toán phân loại:

```
/usr/bin/python3.6 /home/thanhmv/ThanhMai/BAP/SupportVectorMachine/example.py
Loading ./input/test/seg_train/seg_train
Loading ./input/test/seg_test/seg_test
1 541.3046869056801
Accuracy of SVM:  11.71875 %
101 438.99370815244606
Accuracy of SVM:  18.75 %
201 483.2424330939659
Accuracy of SVM:  15.625 %
301 491.0592592608258
Accuracy of SVM:  22.65625 %
401 477.2970209138103
Accuracy of SVM:  14.84375 %
501 439.9411323401143
Accuracy of SVM:  22.65625 %
601 432.37174443131045
Accuracy of SVM:  14.84375 %
701 437.06142192099026
Accuracy of SVM:  23.4375 %
801 445.00604286595404
Accuracy of SVM:  14.84375 %
901 422.35126633766134
Accuracy of SVM:  18.75 %
Accuracy of SVM on data test:  17.17622810030917 %
```

- Kết quả đạt được khoảng 17% → thấp

- Khi kéo các picxel của ảnh thành vector, các đặc trưng về hình khối không còn được xét đến. Việc phân loại dựa vào màu sắc và sự tương quan giữa các hình trong 1 class. Do hình dạng, tương quan giữa các hình trong 1 class rất đa dạng nên việc dùng SVM không có hiệu quả

* Dùng SVM ở lớp cuối cùng thay cho softmax (activation=’linear’, loss=’hinge’):

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import optimizers
from keras.utils import np_utils

train_labels = np_utils.to_categorical(train_labels, 6)
test_labels = np_utils.to_categorical(test_labels, 6)
val_labels = np_utils.to_categorical(val_labels, 6)

model = Sequential()
model.add(Conv2D(16, (3, 3), padding='valid', activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), padding='valid', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dense(6))
model.add(Activation('linear'))
#model.add(Dense(6, activation='softmax'))
model.compile(optimizer='adam', loss='hinge', metrics=['accuracy'])
h = model.fit(train_images, train_labels, batch_size=128, epochs=15, validation_data=(val_imag
es, val_labels))
```

- Cho kết quả tương đương với softmax. SVM ở đây phân loại những features đã được trích xuất vào các class tương ứng nên không gây mất mát .

```
Train on 14034 samples, validate on 3000 samples
Epoch 1/15
14034/14034 [==============================] - 182s 13ms/step - loss: 0.3957 - accuracy: 0.
3587 - val_loss: 0.2972 - val_accuracy: 0.6117
Epoch 2/15
14034/14034 [==============================] - 184s 13ms/step - loss: 0.2940 - accuracy: 0.
5990 - val_loss: 0.2250 - val_accuracy: 0.7167
Epoch 3/15
14034/14034 [==============================] - 195s 14ms/step - loss: 0.2499 - accuracy: 0.
6816 - val_loss: 0.1971 - val_accuracy: 0.7517
Epoch 4/15
14034/14034 [==============================] - 192s 14ms/step - loss: 0.2033 - accuracy: 0.
7521 - val_loss: 0.1727 - val_accuracy: 0.7797
Epoch 5/15
14034/14034 [==============================] - 187s 13ms/step - loss: 0.1752 - accuracy: 0.
7889 - val_loss: 0.1591 - val_accuracy: 0.7863
Epoch 6/15
14034/14034 [==============================] - 197s 14ms/step - loss: 0.1534 - accuracy: 0.
8154 - val_loss: 0.1572 - val_accuracy: 0.8067
Epoch 7/15
14034/14034 [==============================] - 201s 14ms/step - loss: 0.1437 - accuracy: 0.
8308 - val_loss: 0.1123 - val_accuracy: 0.8603
Epoch 8/15
14034/14034 [==============================] - 203s 14ms/step - loss: 0.1250 - accuracy: 0.
8553 - val_loss: 0.0900 - val_accuracy: 0.8833
Epoch 9/15
14034/14034 [==============================] - 201s 14ms/step - loss: 0.1126 - accuracy: 0.
8663 - val_loss: 0.1015 - val_accuracy: 0.8790
Epoch 10/15
14034/14034 [==============================] - 208s 15ms/step - loss: 0.1058 - accuracy: 0.
8769 - val_loss: 0.0987 - val_accuracy: 0.8753
Epoch 11/15
14034/14034 [==============================] - 204s 15ms/step - loss: 0.0934 - accuracy: 0.
8909 - val_loss: 0.0764 - val_accuracy: 0.9120
Epoch 12/15
14034/14034 [==============================] - 199s 14ms/step - loss: 0.0917 - accuracy: 0.
8906 - val_loss: 0.0780 - val_accuracy: 0.9063
Epoch 13/15
14034/14034 [==============================] - 195s 14ms/step - loss: 0.0852 - accuracy: 0.
9025 - val_loss: 0.0601 - val_accuracy: 0.9267
Epoch 14/15
14034/14034 [==============================] - 207s 15ms/step - loss: 0.0732 - accuracy: 0.
9144 - val_loss: 0.0675 - val_accuracy: 0.9140
Epoch 15/15
14034/14034 [==============================] - 213s 15ms/step - loss: 0.0646 - accuracy: 0.
9263 - val_loss: 0.0539 - val_accuracy: 0.9327
```

```
score = model.evaluate(test_images, test_labels, verbose=0)
print(score)
```

```
[0.11554687689741452, 0.8586666584014893]
```