In [7]:
```
import exercise2_config as config
import readers

train_images = readers.ReadFromCsvFile(config.TRAIN_DATA_FILE)
test_images = readers.ReadFromCsvFile(config.TEST_DATA_FILE)
```

In [8]:
```
train_data = [image.pixels for image in train_images]
train_labels = [image.label for image in train_images]
test_data = [image.pixels for image in test_images]
test_labels = [image.label for image in test_images]
```

In [9]:
```
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

We tried to run the program on the entire dataset, but it wouldn't finish after more than 8 hours. Therefore, we decided to test it on a subset of 5000 entries.

In [10]:
```
# To be removed when analyzing the entire set

train_data = train_data[:5000]
train_labels = train_labels[:5000]
test_data = test_data[:5000]
test_labels = test_labels[:5000]
```

We first use a linear kernel and optimize the C parameter by using a 5-fold cross-validation. We deided on a one versus one classifier. The best value for the C parameter as well as the average accuracy are displayed below.

In [13]:
```
kernel_param = 'linear'

validation_means = []
for k in range(-5, 5):
    C_param = 10**k
    classifier = svm.SVC(kernel=kernel_param, C=C_param, decision_function_shape='ovo')
    # linear kernel (default would be rbf)
    # one vs one approach

    scores = cross_val_score(classifier, train_data, train_labels, cv=5, n_jobs=3)  # Cross validation of the t
    tuple_mean = (C_param, scores.mean())
    validation_means.append(tuple_mean)
validation_means.sort(key=lambda a: a[1])
best_C = validation_means[-1][0]
best_mean = validation_means[-1][1]
print(f"The best value for the C parameter is {best_C} with an accuracy of {best_mean} for the 5-fold CV.")
```

```
The best value for the C parameter is 1e-05 with an accuracy of 0.9077999999999999 for the 5-fold CV.
```

Now that we have the optimized value for the C parameter, we build our SVM and classify the test set. The accuracy of our classifier is displayed below.

In [14]:
```
classifier = svm.SVC(kernel=kernel_param, C=best_C, decision_function_shape='ovo')
classifier.fit(train_data, train_labels)
predicted_class = classifier.predict(test_data)  # Predict the class
print(f"Accuracy of our classifier : {accuracy_score(test_labels, predicted_class)}")
```

```
Accuracy of our classifier : 0.8818
```

Now we can try the same thing but with another kernel. We used here the rbf.

In [11]:
```
kernel_param = 'rbf'

validation_means = []
for k in range(-5, 5):
    for l in range(-7, 3):
        C_param = 10**k
        gamma_param = 10**l
        classifier = svm.SVC(kernel=kernel_param, gamma=gamma_param, C=C_param, decision_function_shape='ovo')
        # rbf kernel
        # one vs one approach

        scores = cross_val_score(classifier, train_data, train_labels, cv=5, n_jobs=3)  # Cross validation of t
        tuple_mean = (C_param, gamma_param, scores.mean())
        validation_means.append(tuple_mean)
validation_means.sort(key=lambda a: a[2])
best_C = validation_means[-1][0]
best_gamma = validation_means[-1][1]
best_mean = validation_means[-1][2]
print(f"The best pair of parameters is C = {best_C} and gamma = {best_gamma} with an accuracy of {best_mean} fo
```

```
The best pair of parameters is C = 10 and gamma = 1e-07 with an accuracy of 0.9494 for the 5-fold CV.
```

In [12]:
```
classifier = svm.SVC(kernel=kernel_param, gamma=best_gamma, C=best_C, decision_function_shape='ovo')
classifier.fit(train_data, train_labels)
predicted_class = classifier.predict(test_data)  # Predict the class
print(f"Accuracy of our classifier : {accuracy_score(test_labels, predicted_class)}")
```

```
Accuracy of our classifier : 0.9316
```

We can see a difference between the two kernels over these data. The rbf is 5% better than the linear.