CNN for skin cancer detection

장림초
AD20206806

Mole Classifier Kernel

Skin cancer is the most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions.

피부암은 인간의 악성종양 중 가장 흔한 것으로, 주로 초기 임상검사로 시작하여 진피경 분석, 조직검사 및 조직병리학적 검사에 따라 시각적으로 진단된다. 이미지를 사용한 피부 병변의 자동 분류는 피부 병변 외관의 미세한 변동성으로 인해 어려운 작업이다.

Dataset

- The dataset is taken from the ISIC (International Skin Image Collaboration) Archive.
  데이터 세트는 ISIC(International Skin Image Collaboration) 아카이브에서 가져왔다.

Dataset:

- 1800 pictures of benign moles
- 1497 pictures of malignant moles
- Resized (224x224x3) RGB

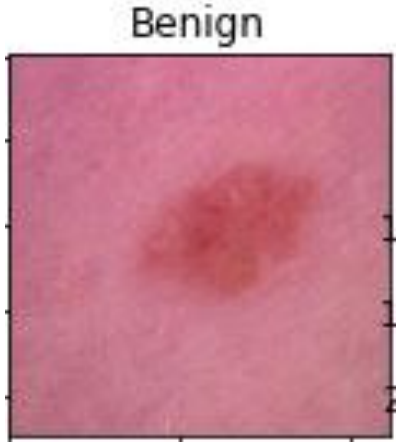
The task of this kernel is to create a model, which can classify a mole visually into benign and malignant.
이 커널의 임무는 두더지를 시각적으로 양성과 악성으로 분류할 수 있는 모델을 만드는 것입니다.

Task and Classification

1. Create model --- kernel
2. The model will be tested on the accuracy score

2 different classes of skin cance:

- **Benign**
- **Malignant**



In this kernel I will try to detect 2 different classes of moles using Convolution Neural Network with keras tensorflow in backend and then analyse the result to see how the model can be useful in practical scenario.

이 커널(kernel)에서 백엔드에 keras tensorflow 가 있는 Convolution Neural Network 을 사용하여 두 가지 등급의 몰을 탐지한 다음 결과를 분석하여 모델이 실제 시나리오에서 어떻게 유용할 수 있는지 확인하려고 한다.

Model Building and Evaluation

Step 1 : Importing Essential Libraries

Step 2: Loading pictures and making Dictionary of images and labels

Step 3: Categorical Labels

Step 4: Normalization

Step 5: Train and Test Split

Step 6: Model Building

Step 7: Cross-validating model

Step 8: Testing model

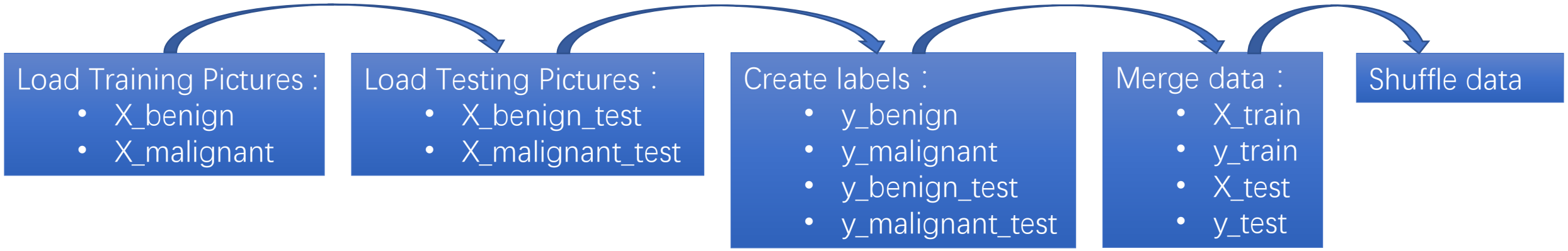
Step 1 : Importing Essential Libraries

Using TensorFlow backend:

```
%matplotlib inline    #Drawing

import numpy as np       # Computing library

import pandas as pd      #Data manipulation and analysis

import keras      #Library

.....
```

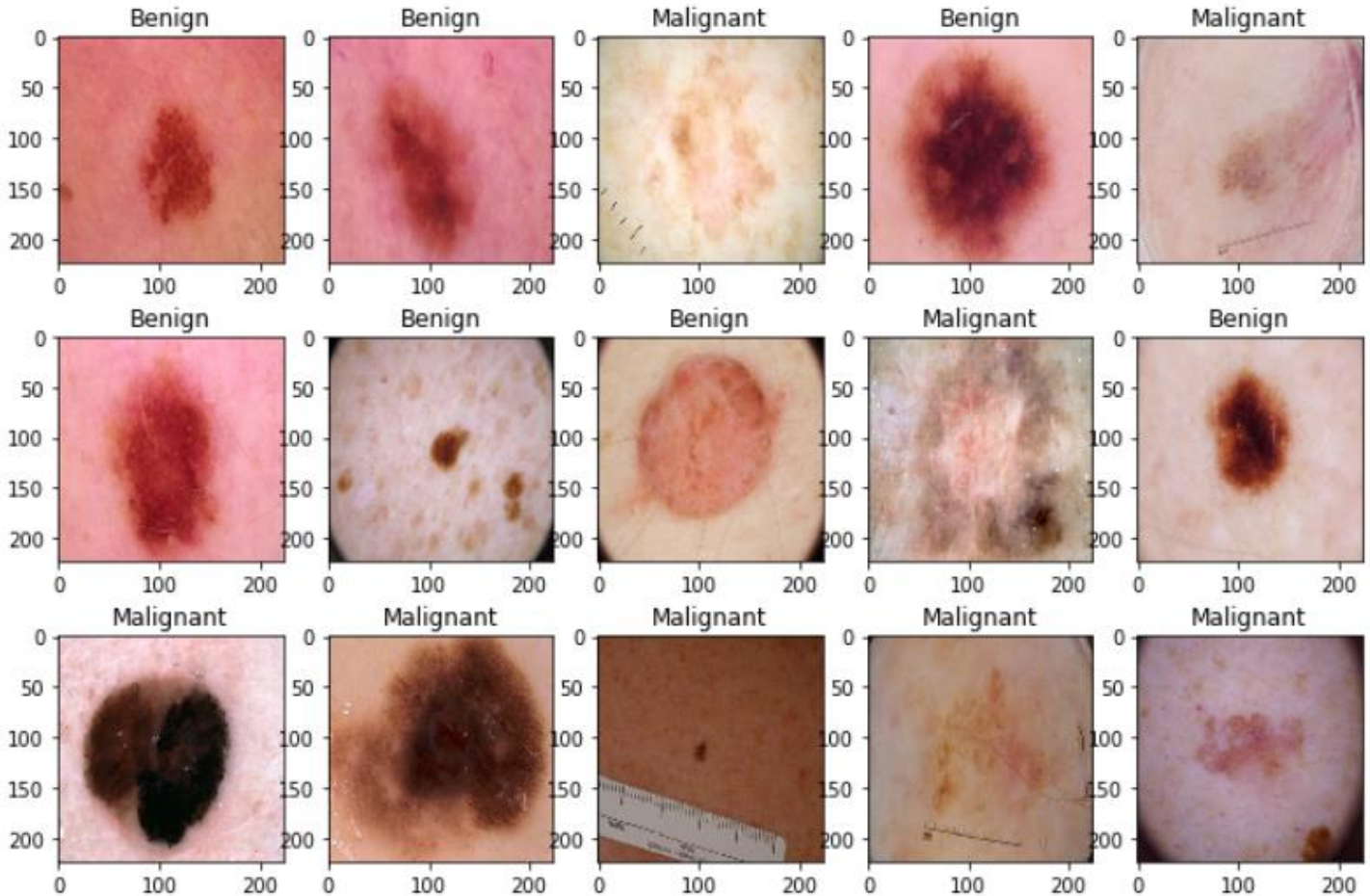
Step 2: Loading pictures and making Dictionary of images and labels

1. In this step load in the pictures and turn them into numpy arrays using their RGB values.
   이 단계에서는 사진을 로드하고 RGB 값을 사용하여 numpy 배열로 바꿉다.
2. Create label---- As the pictures do not have any labels, these need to be created.
   그림에는 레이블이 없으므로 이러한 레이블을 만들어야 한다.
3. Finally, the pictures are added together to a big training set and shuffeled.
   마지막으로, 이 사진들은 큰 훈련 세트에 합쳐지고 섞이다.



Step 2: Loading pictures and making Dictionary of images and labels

- Display first 15 images of moles
- Classified result



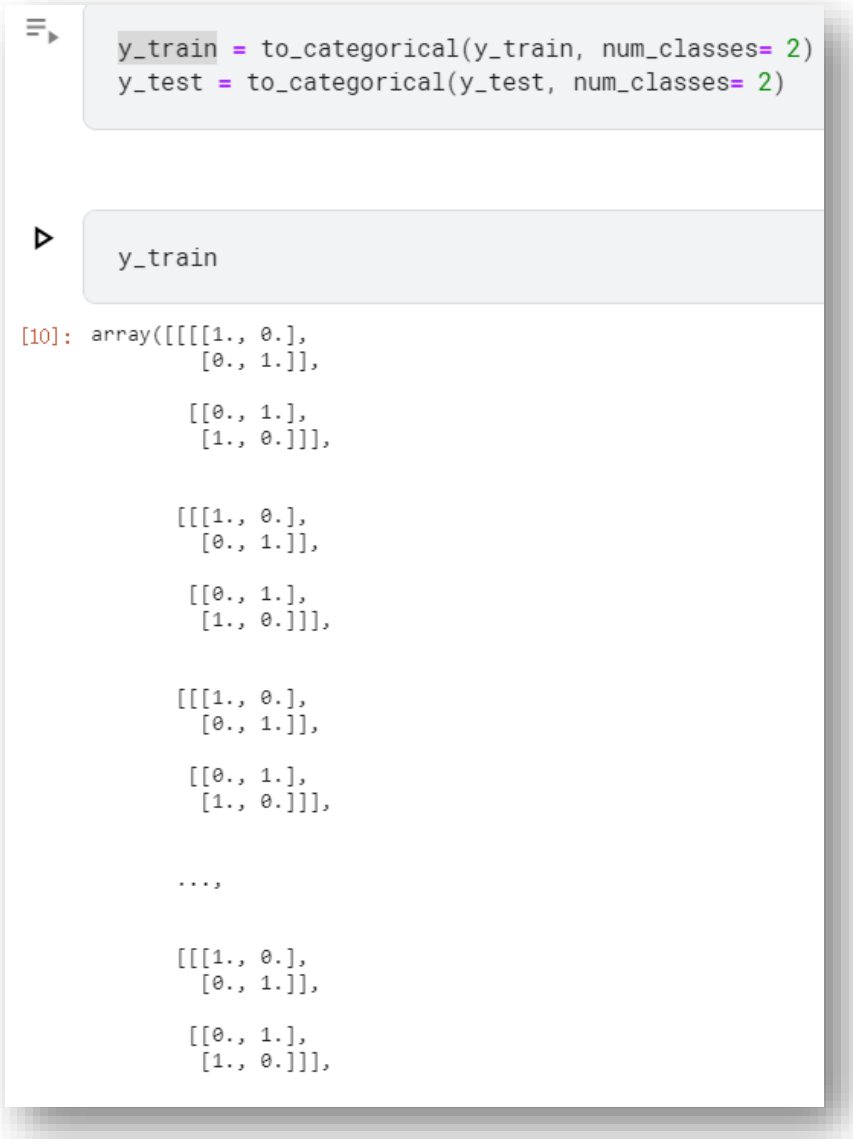
Step 3: Categorical Labels

```
tf.keras.utils.to_categorical(
    y, num_classes=None, dtype='float32'
)
```

- Converts a class vector (integers) to binary class matrix.

클래스 벡터(정수)를 이진 클래스 행렬로 변환합니다.

- Turn labels into one hot encoding



Step 4: Normalization

목적 :

1. Standardize each dimension of the feature to a specific interval
   특징의 각 측면을 특정한 구간까지 표준화하다
2. Change a dimensional expression to a non-dimensional expression
   차원 표현식을 무차원 표현식으로 변경

장점:

1. Speed up the convergence speed of models based on gradient descent or stochastic gradient descent
   사다리 하강법 또는 무작위 사다리 하강법 모델에 기초한 수렴 속도 가속화
2. Improve the accuracy of the model 모형의 정밀도를 높이다.

- Normalize all Values of the pictures by dividing all the RGB values by 255

  모든 RGB 값을 255로 나누어 그림의 모든 값을 정규화한다.

- With data augmentation to prevent overfitting

  과적합을 방지하기 위한 데이터 확대 기능 포함

```
# With data augmentation to prevent overfitting
X_train = X_train/255.
X_test = X_test/255.
```

+ Code + Markdown

```
X_train
```

```
array([[[[218, 122, 133],
         [215, 119, 131],
         [216, 123, 134],
         ...,
         [215, 122, 133],
         [209, 116, 126],
         [212, 114, 127]],

        [[217, 120, 131],
         [212, 115, 126],
         [218, 121, 132],
         ...,
         [217, 120, 139],
         [212, 116, 128],
         [210, 113, 122]],

        [[220, 128, 139],
         [215, 122, 132],
         [217, 120, 131],
         ...,
         [216, 117, 135],
         [216, 120, 132],
         [212, 116, 127]],

        ...,
```

Step 5: Train and Test Split

train_test_split() :

- Divide the original data into "test set" and "training set" in proportion.
  원본 데이터를 ' test set '과 ' training set '으로 비례해 나눈다.
- Return the divided samples and labels.

X_train,X_test, y_train, y_test = cross_validation.train_test_split**(** **train_data**, **train_target**, **test_size**=0.3, random_state=0)

train_data: Sample feature set to be divided (구분된 표본 특징집)

train_target: Sample label to be divided(구분된 샘플 라벨)

test_size: Indicates the proportion of samples(표본 비율)

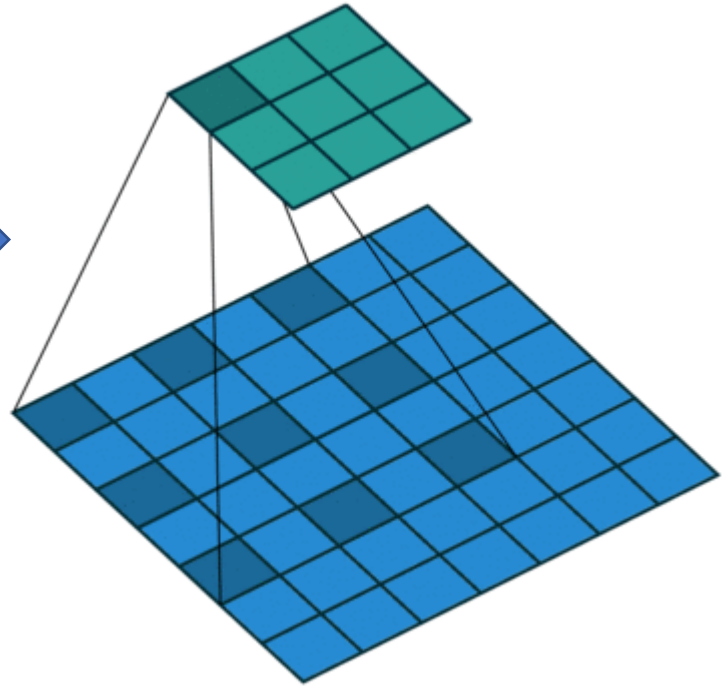
random_state: Random number

Step 6: Model Building - Conv2D

Keras Sequential Model: Use API

First layer -- convolutional (Conv2D) layer

```
Conv2D(64, kernel_size=(3, 3),padding = 'Same',input_shape=input_shape,
activation= activ, kernel_initializer='glorot_uniform')
```



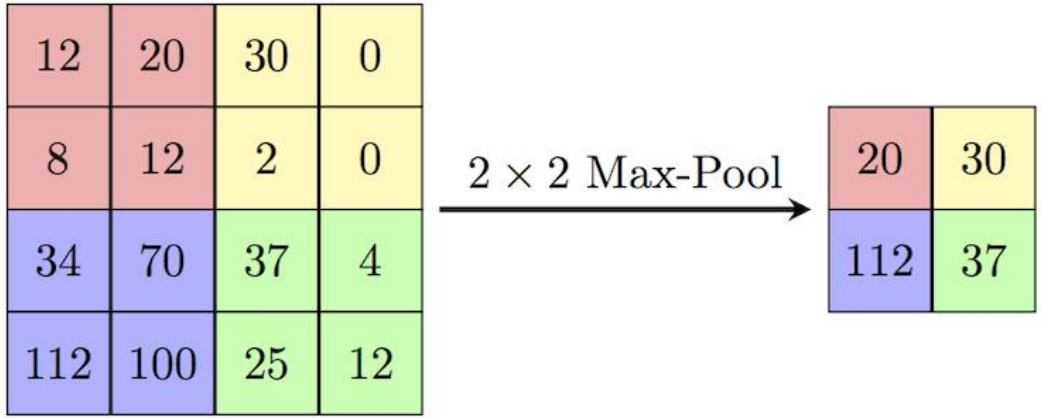
1. choosed to set 64 filters for the two firsts conv2D layers.
2. Each filter transforms a part of the image using the kernel filter.
3. The kernel filter matrix is applied on the whole image.
4. Filters can be seen as a transformation of the image.

1. 두 개의 첫 번째 conv2D layers 에 대해 64개의 필터를 설정하도록 선택되었다.
2. 각 필터는 커널 필터를 사용하여 이미지의 일부를 변환하다.
3. 커널 필터 매트릭스가 전체 이미지에 적용되다.
4. 필터는 이미지의 변환으로 볼 수 있다.

Step 6: Model Building - MaxPool2D

Second layer -- Pooling (MaxPool2D) layer

```
MaxPool2D(pool_size = (2, 2))
```



1. Looks at the 2 neighboring pixels and picks the maximal value.
   인접한 두 픽셀을 보고 최대값을 선택하다.
2. Reduce computational cost, and to some extent also reduce overfitting.
   계산 비용을 줄이고 과적합도 어느 정도 줄이다.
3. pool_size: more the pooling dimension is high, more the downsampling is important.
   풀링 차원이 높을수록 다운샘플링이 중요하다.

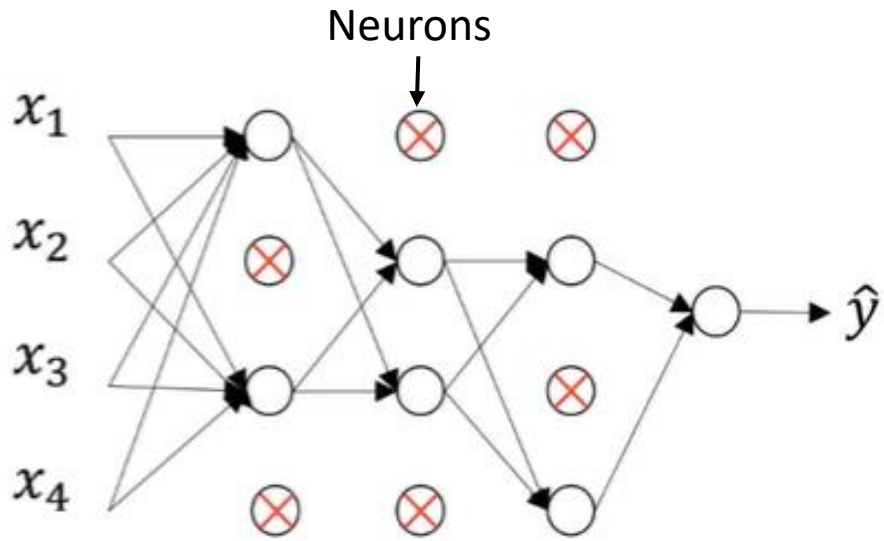
Step 6: Model Building - Dropout

Third layer -- Dropout layer:

Structure used to reduce neural network overfitting

신경망 과적합을 줄이는 데 사용되는 구조

```
Dropout(0.25)
```



1. Randomly drop some neurons during training.

   훈련 중에 몇 개의 neurons 을 무작위로 떨어뜨리다.

2. Forces the network to learn features in a distributed way.

   네트워크가 분산 방식으로 기능을 학습하도록 하다.

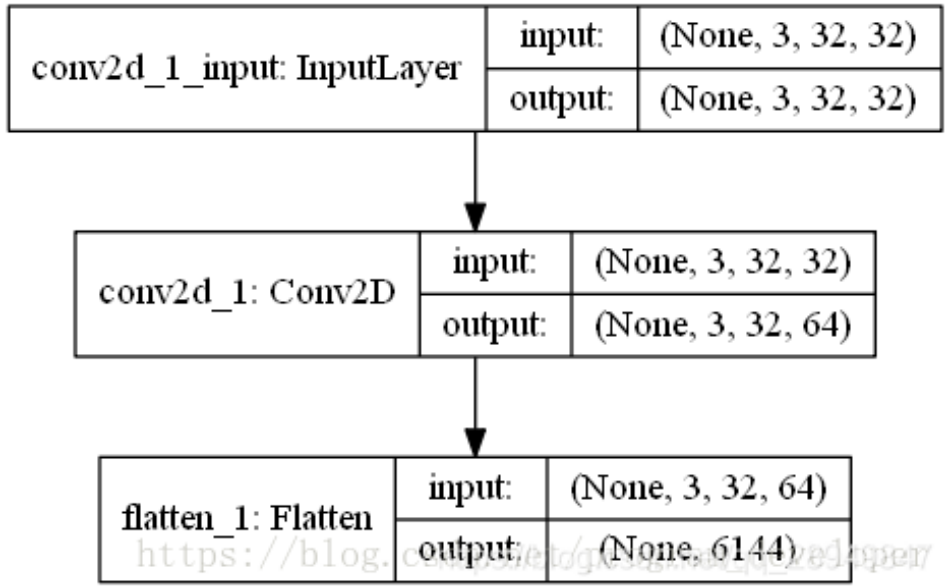
Step 6: Model Building - Flatten

Flatten layer :

Multidimensional input unidimensionalization

다차원 입력의 1차원화

Flatten ()



1. Convert the final feature maps into a one single 1D vector.

   최종 기능 맵을 하나의 단일 1D 벡터로 변환한다..

2. Make use of fully connected layers after some convolutional/maxpool layers.

   일부 컨볼루션/맥스풀 레이어 후에 완전히 연결된 레이어를 사용하다.

3. Combines all the found local features of the previous convolutional layers.

   이전 컨볼루션 layers 의 발견된 모든 로컬 특징을 결합하다.

Step 7: Cross-validating model

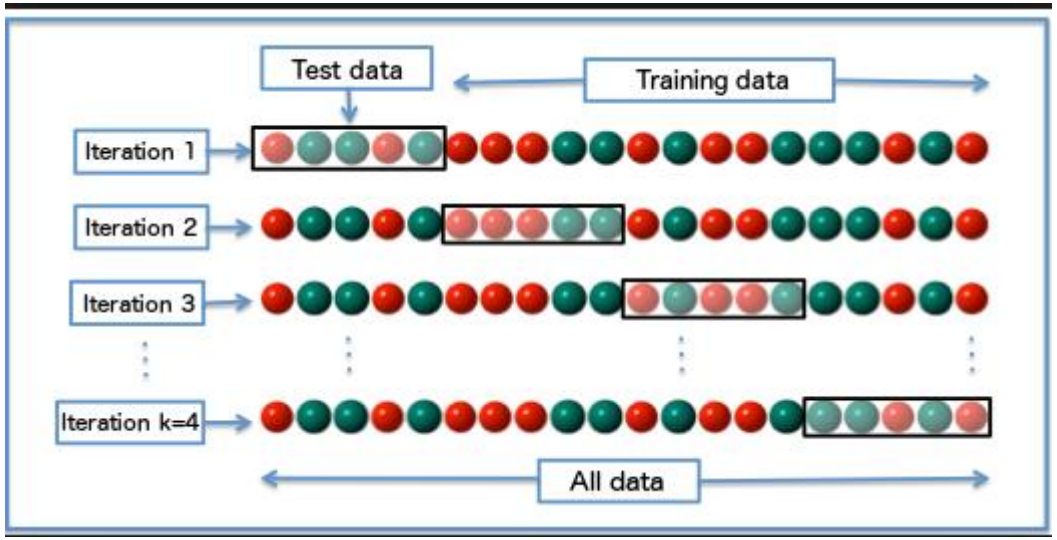
Cross validation (CV) is one of the technique used to test the effectiveness of a machine learning models, it is also a re-sampling procedure used to evaluate a model if we have a limited data.

Cross validation (CV) 는 기계 학습 모델의 효과를 테스트하기 위해 사용되는 기법 중 하나이며, 제한된 데이터가 있는 경우 모델을 평가하는 데 사용되는 재샘플링 절차이기도 하다.

1.Train_Test Split approach (훈련 테스트 분할 접근 방식)

2.K-Folds Cross Validation (K-Folds 교차 검증)(This article uses this method)

Step 7: Cross-validating model



K-Folds Cross Validation (K-Folds 교차 검증)

1. Split the entire data randomly into K folds
2. the model using the K-1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.
3. Repeat this process until every K-fold serve as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

Step 8: Testing model

```
# Fitting model to all data
model = build(lr = lr,
              init = init,
              activ = activ,
              optim = optim,
              input_shape = input_shape)

model.fit(X_train, y_train,
              epochs = epochs, batch_size = batch_size, verbose = 0,
              callbacks = [learning_rate_reduction]
)

# Testing model on test data to evaluate
y_pred = model.predict_classes(X_test)

print(accuracy_score(np.argmax(y_test, axis = 1), y_pred))
```

Thank you