빈페이지**목차**

1. **Introduction**
2. 주제명 / 주제에 대한 개요
3. What is CNN?
4. Source Code
5. **Modeling**
6. Data Augmentation
7. How to make CNN modelS
8. **Classification**
9. Data Augmentation
10. Softmax / One-hot encoding
11. **Auto Coloring**
12. 영역별 색이 칠해지는 조건
13. **Analysis & Conclusion**
14. Analyze hypherparameters
15. Hypherparameter 와 loss value 의 상관관계
16. Layer 재구성을 통한 Model의 재정의
17. 아쉬운 점
18. **Reference**
19. **Introduction**
20. **프로젝트 주제**

CNN을 이용하여 단색(검정색) 선으로 구성된 과일 이미지를 정해진 label에 따라 classification을 진행하고 분류한 label에 따라 과일을 자동으로 채색한다.

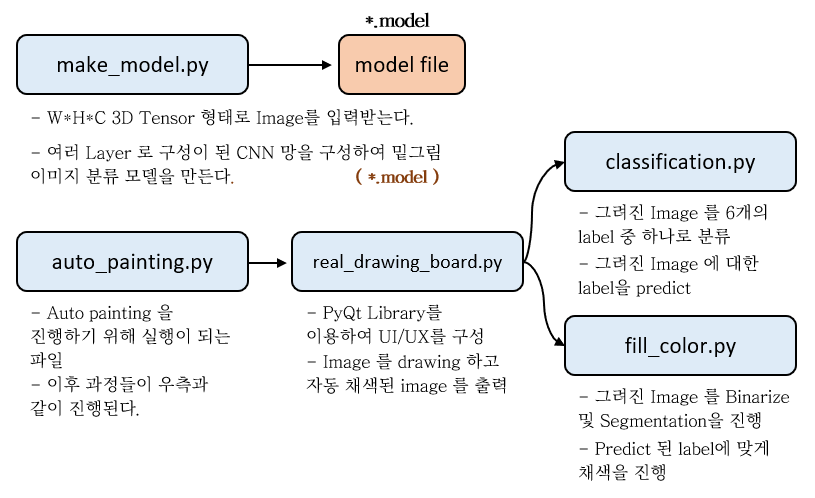
1. **What is CNN?**

CNN(Convolutional Neural Network, 합성곱 신경망)은 시신경 구조를 모방해 만든 기술이다. 기존의 Fully Conneccted Layer로 구성된 Neural Network는 Data를 Vector로 처리하여 Image data의 경우 인근 Pixel과의 관계나 Pattern등 공간 정보를 고려하기 어렵다는 단점이 있었다.

그러나, CNN은 Data를 Vector화가 아닌 3D Tensor로 처리하여 공간 정보를 유지하고 Filter를 이용해 Feature map을 생성하고 학습할 수 있다. 이 때문에 Computer Vision 분야에서 우수한 성능을 보인다.

본 프로젝트는 이러한 CNN의 특성을 이용하여 직접 그린 과일 Image를 분류하고, 추가적으로 Image의 각 영역을 구분하여 자동으로 특정 색으로 채색을 진행하는 프로그램을 구현한다.

1. **Source Code**



1. **Modeling**
2. **Data Augmentation**

* Image resizing

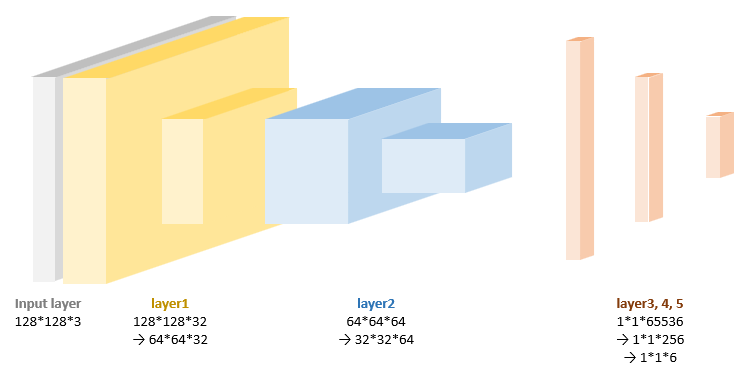
Train을 위한 image data들과 Sketched image data 들을 모두 128\*128 pixel 로 resizing 을 진행하였다. 64\*64 pixel로 resizing 하는 것과 128\*128 pixel로 resizing 하는 것중 더 세부적인 training을 위해 128\*128 pixel의 image로 resizing하는 방향으로 진행하였다.

* Image rotation

우선 [‘apple’, ‘tomato’, ‘orientalmelon’, ‘watermelon’, ‘carrot’, ‘strawberry’] 6개의 각각의 label 에 대해서 직접 그린 100개의 data를 통해 train\_data를 확보하였다. 이 Image data들을 모두 하나의 (W\*H\*Channel) 형태를 가진다. 그 후 Image rotation 으로 모든 Image들을 -20도 단위로 rotate한 image들도 Train시에 사용하였다.

1. **How to make CNN model**

* Default CNN model



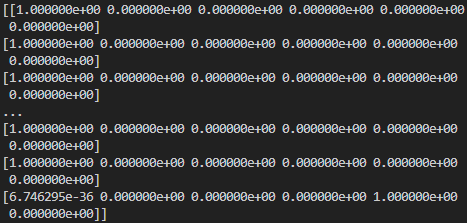
기존의 model은 위와 같이 5-layer model로 구성이 되어있다. 모든 Convolution layer 들은 3\*3 size filter와 padding을 진행하여 공간 정보를 유지하면서 channel의 개수를 2배 늘려가는 방식으로 구현되었다. 그리고 Convolution layer들을 지난 후 항상 maxPooling을 활용하여 공간 정보를 1/2로 낮추고 1/4 확률로 dropout을 진행한다.

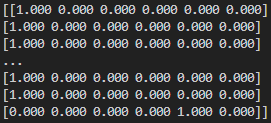
위 작업이 끝난 뒤 flatten을 시킨 후 256 length의 vector로 바꾸고 이어서 output을 도출할 수 있게끔 class의 수 만큼 vector의 length를 변경한다. 이 때 마지막 Dense method 에서 softmax activation function을 사용하여 각 6개의 label들일 확률 값으로 변경을 한다. 이 softmax를 통해 output된 vector는 classification.py에서 확인할 수 있다.

1. **Classification**
2. **Data Augmentation**

Modeling을 진행하는 과정에서 Resizing을 하는 Data Augmentation을 진행 했으므로 동일한 조건을 위해 Sektch된 image도 128\*128 pixel로 resizing을 진행한다.

1. **Softmax**



**→** 

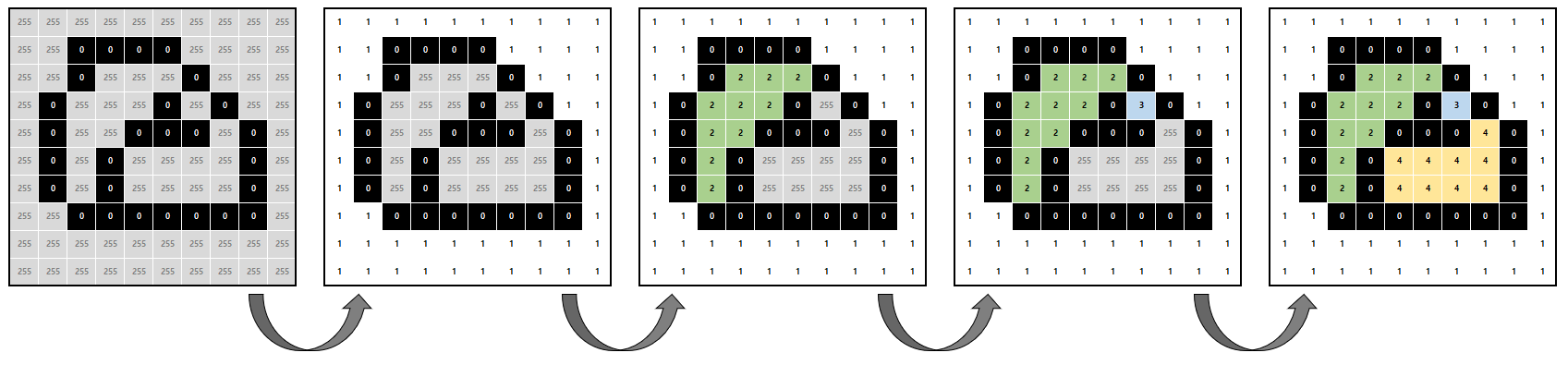
좌측 경우는 Model을 통해 예측이 된 label vector들이다. Model이 마지막에 output vector를 출력할 때 Softmax activation function을 이용하였기 때문에 각 label들에 해당되는 값들은 그 label일 확률에 해당이 된다. Sketch한 그림 하나에는 6차원 vector가 출력되며, 좌측의 경우는 지금까지 sketch된 그림들의 label vector를 모두 출력 한 것 이다. 좌측과 같이 정의가 된 output vector의 모든 element값들을 우측과 같이 소수 셋째자리 까지 반올림 한다.

그 결과 가장 마지막 vector의 첫 번째 element 였던 6.746295e-36 값은 0이 되고 더 다루기 쉬운 vector의 형태를 띄는 것을 알 수 있다.

최종적으로 위 label vector에서 0.8이상의 값을 띄는 label이 현재 입력 data의 predicted label이 된다.

1. **Auto Coloring**

Coloring을 진행하는 과일은 두 부류로 나뉜다. 두 가지 종류의 색을 칠하는 과일과 한 가지 색만 칠하는 과일이 있다. 사용한 기존의 Open source의 class및 method를 수정하여 label에 따라 유동적인 종류의 색을 칠할 수 있는 함수를 구현하였다.



Image는 위와 같은 방식으로 count값을 1부터 부여하면서 같은 count값을 가진 pixel들에 같은 색을 칠하는 방식으로 coloring이 진행된다.

1. **Analysis & Conclusion**
2. **Augmented data를 활용한 model의 효율성 확인**

* <batch 32, 64, 128, 256 인 경우 체크>

1. **Augmented data를 활용하지 않은 model을 통한 효율성 확인**

Augmented data를 활용하여 modeling을 진행하면, 한 epoch당 2분 정도가 소요되어 구현하고자 하는 epoch의 수 (50~100)를 구현하기에 어려움이 있었다. 따라서 Augmented data를 제외한 600개의 Plain train data만 활용하여 각 Hyperparameter(optimizer, batch, epoch, learning\_rate)를 바꾸어 보면서 효율성을 비교하였다.

* Optimizer 비교 (Adam, Nadam)

<adam nadam 설명>

|  |  |  |
| --- | --- | --- |
|  | **Adam** | **Nadam** |
| acc |  |  |
| loss |  |  |

기존의 Adam Optimizer를 Nadam Optimizer로 바꿔보았다. Batch size는 128로 50개의 Epoch를 실행해 본 결과 위와 같았다. 그 결과 기존의 Adam Optimizer를 사용한 것 보다 Nadam Optimizer를 사용한 경우 validation data set의 accuracy, loss 값이 Adam optimizer를 사용한 경우보다 더 성능이 좋은 것을 확인하였다. 따라서 이후 여러 Hyperparameter들을 바꿔보면서 효율성을 체크하는 과정에 Nadam optimizer를 사용하였다.

* Learning rate 비교

1. **References**

* [CNN을 통한 구름 분류 (tistory.com)](https://junseokk.tistory.com/4)
* [컨볼루션 신경망 모델 만들어보기 (tykimos.github.io)](https://tykimos.github.io/2017/03/08/CNN_Getting_Started/)
* [googlecreativelab/quickdraw-dataset: Documentation on how to access and use the Quick, Draw! Dataset. (github.com)](https://github.com/googlecreativelab/quickdraw-dataset#get-the-data)
* [데이터 증강  |  TensorFlow Core](https://www.tensorflow.org/tutorials/images/data_augmentation?hl=ko)
* [3.13. 드롭아웃(dropout) — Dive into Deep Learning documentation (d2l.ai)](https://ko.d2l.ai/chapter_deep-learning-basics/dropout.html)
* [컨볼루션 신경망 레이어 이야기 (tykimos.github.io)](https://tykimos.github.io/2017/01/27/CNN_Layer_Talk/)
* [tf.keras.callbacks.LearningRateScheduler (tistory.com)](https://dodonam.tistory.com/178)