CNN을 이용한 아시아 고궁 분류

(Classification for Ancient Palace in asia using Convolutional Neural Networks )

김주환, 최우혁, 임바울

고려대학교 세종캠퍼스 과학기술대학 전자 및 정보공학과, 세종시 2511, 고려대학교

**ABSTRACT**

본 연구는 인공지능 분야의 딥러닝 기술을 바탕으로 아시아 고궁 사진을 분류 시스템을 구연하였다. 시스템 성능 극대화를 위해서 data 수집 시 생긴 중복된 사진 및 워터마크가 있는 사진을 재가공하고, Convolutional Neural Network(CNN)기반의 VGG16 model을 사용 했으며, 결과를 위해 128x128, 256x256 size의 한국,중국,일본 사진을 model에 학습시키면서,가장 적합한 model의 accuracay를 찾는 실험을 통해 data를 전처리한 256x256size 사진의 결과가 성능이 좋은 것을 알 수 있었다.

**Keywords**: VGG16, Convolutional Neural Network, 아시아 고궁 사진을 분류 시스템, model accuracy

1. **서론**

21세기 글로벌 경제 시대의 산업에 있어서 인공지능을 기반으로 하는 4차 산업 혁명이 관광산업에서도 큰 영향을 미치고 있다. 현재 코로나 사태로 여행이 자유롭지 않지만 밀레니얼 세대로 인해 여행을 자주 다니며 여행에 대해 많은 정보를 얻을 수 있는 세상에 살아가고 있다. 이러한 이유로 이번 논문에서는 여행지 사진만을 보고 어떤 나라인지 판단해 여행지 를 선정하고, 그 중 아시아 고궁 사진을 보고 어떤 나라인지 판단해 여행지 선정에 도움이 되고자 한다.

1. **본론**
2. 데이터

아시아 고궁을 주제로 CNN model에 학습하기 위해서는 data를 설정한 이유는, 각 나라별 성의 특징이 비슷한 점이 많다는 것이다. 이 점을 고려하여 3개의 data set을 가지고 실험을 진행 했다.

첫번째 data set은 중국, 일본, 한국 고궁 사진을 각 class 별로 800장씩 data를 나누어 학습시켰으며, 똑같은 주제의 사진이지만 구도가 조금만 달라도 data에 포함시켰다.

두번째 data set은 이전의 800장 data 사진에서 사진 수를 줄여 각 class 별로 500장씩 data를 나누어 학습시켰으며, 이번에는 고궁 구도가 정면, 측면, 후면 기준 인 것을 포함시켰다.

세번째 data set은 이전 data set에서 사진의 수를 더 줄여 data 100장을 학습시켰으며, 고궁 구도가 정면 그리고 서로 다른 고궁 사진을 정리하여 학습시켰다.

1. 모델

VGG16 model에서 MLP를 추가하여, flatten layer, Dropout layer, Batch Normalization을 차례대로 추가하여 실험을 진행하고, output layer의 activation function은 sigmoid, softmax 함수를 쓰면서 predict accuracy가 좋은 것을 결과로 사용 했다.

2 - (1) VGG16

VGGNet은 옥스포드 대학의 연구팀VGG에 의해 개발된 모델로써, 여기서 말하는 VGGNet은 16개 또는 19개의 layer로 구성된 model을 의미 하며, VGG16, VGG19이라고 한다.

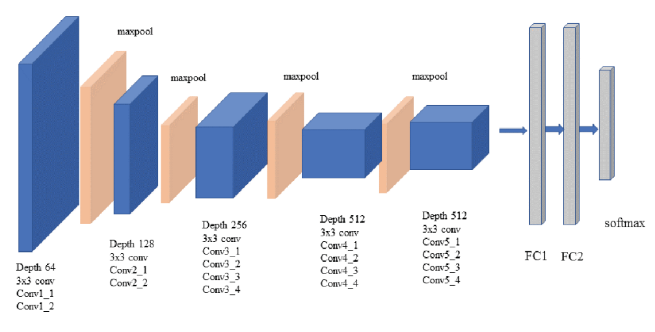
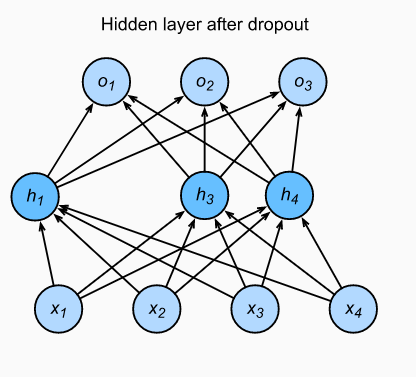


그림1. VGG16 model

2 - (2) Dropout

Dropout이란 hidden neuron일부를 생략하여 학습을 진행하면 생략한 hidden neuron은 학습에 영향을 끼치지 않게 된다. 이 효과를 사용하는 이유는 overfitting 문제와 regularization을 위해서 사용한다.



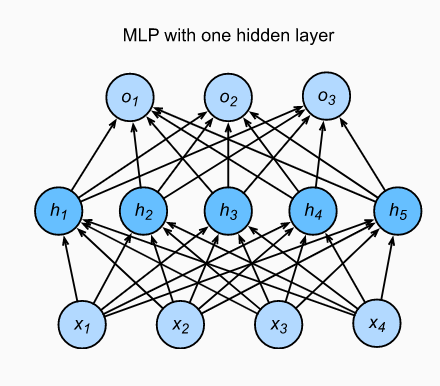


그림 2 MLP Dropout 적용

2 - (3) Batch Normalization

Batch Normalization은 기본적으로 Gradient Vanishing, Gradient Exploding이 일어나지 못하게 하는 방법으로 trainning하는 과정 자체를 전체적으로 안정화하여 학습 속도를 높이기 위해 사용했다.

1. **실험**

dataset은 아래의 표1과 같이 data를 각 class별로 사진의 개수를 다르게 하여 실험을 진행했다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Data800** | **Data500** | **Data100** |
| **train** | **1440** | **900** | **180** |
| **test** | **480** | **300** | **60** |
| **validation** | **480** | **300** | **60** |

**표 1.**

input shape = (128,128,3)로 설정하고, VGG16 model에서 Flatten layer에 Dropout을 추가하여 실험을 진행했으며, output layer activation=‘sigmoid’로, optimizer= optimizer.RMSprop=(lr=1e-4)로 설정하고 data800, data500, data100을 학습 시켜보았다.

다음은 input shape = (128, 128, 3)로 설정하고,

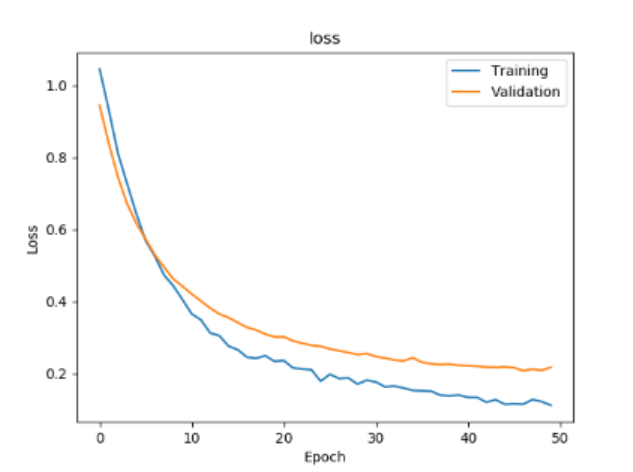
VGG16 model에서 output\_activation=’sigmoid’

로 Optimizer=.RMSprop=(Ir=1e-4)로 설정한 후 실험한 결과이다

|  |  |  |
| --- | --- | --- |
| **Data800**  **Epochs50** | **VGG16**  **Flatten**  **dropout** | **VGG16**  **Flatten**  **Dropout**  **(no\_argument)** |
| **Train acc** | **0.9576** | **0.9958** |
| **Train loss** | **0.1636** | **0.0634** |
| **Train acc** | **0.9416** | **0.9566** |
| **Train loss** | **0.1978** | **0.1538** |

처음에 Data 800의 결과를 비교하면

Argumentation을 하지 않았을 때의 결과가 더 좋았다.



Loss의 결과는 좋았지만 데이터 자체가 비슷한

경우가 많았고 하나의 성을 외워 버리는 경향

이 생겼기 때문에 predict의 결과는 좋지 않았

다.

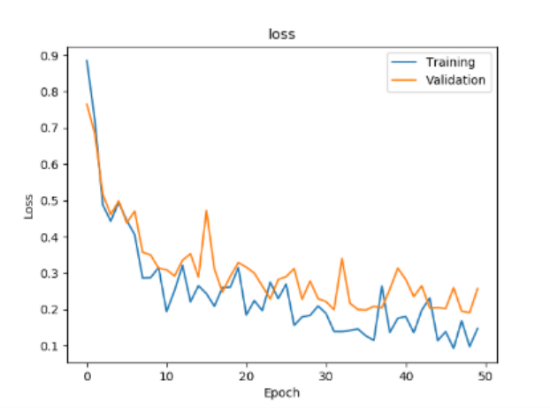
Data 500과 비교했을 때 dropout이 있을

경우 Flatten과 dropout이 있는 결과가 더

좋았다..

|  |  |  |  |
| --- | --- | --- | --- |
| **Data500**  **Epochs50** | **VGG16**  **Flatten** | **VGG16**  **Flatten**  **dropout** | **VGG16**  **Flatten**  **Dropout**  **(no\_argument)** |
| **Train acc** | **0.9766** | **0.9933** | **0.9744** |
| **Train loss** | **0.0736** | **0.0147** | **0.0929** |
| **Train acc** | **0.9166** | **0.9466** | **0.9466** |
| **Train loss** | **0.1868** | **0.2155** | **0.1589** |

Predict는 Data 800과 달리 좋지않았다.



Loss는 Data 800보다 커졌다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data100**  **Epochs50** | **VGG16**  **Flatten** | **VGG16**  **Flatten**  **dropout** | **VGG16**  **Flatten**  **Dropout**  **(no\_argument)** |
| **Train acc** | **1.0** | **1.0** | **1.0** |
| **Train loss** | **0.0050** | **0.0058** | **0.0083** |
| **Train acc** | **0.8666** | **0.85** | **0.8166** |
| **Train loss** | **0.5632** | **0.5422** | **0.6423** |

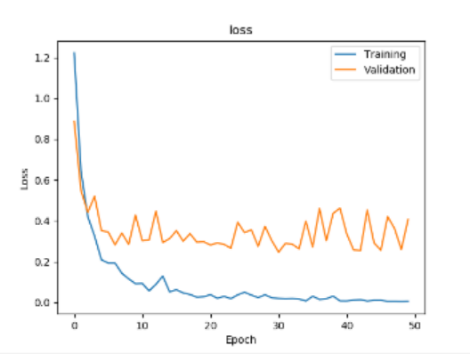
데이터가 많은 경우 dropout을 추가해서 데이터의 결과가 좋았지만, 데이터의 개수가 적을 경우 dropout이 없을 때 결과가 더 좋았다.

비슷한 데이터가 많은 경우 dropout을 추가함으로써 필요한 정보를 더 잘 가져와 결과가 좋았지만 predict는 좋지 않았다.

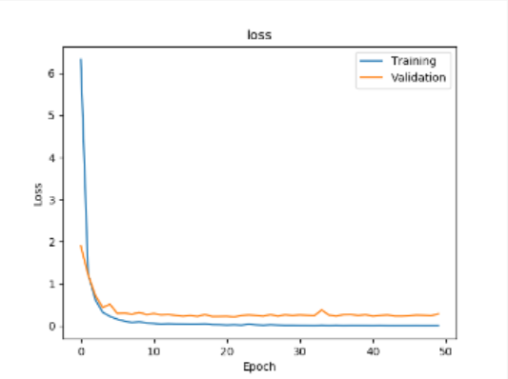
다음은 input shape = (128, 128, 3)로 설정하고,

VGG16 model에서 output\_activation=’softmax로 Optimizer=adam, input\_size(128,128,3)로 설정했을 때 Input\_size (256,256,3)와 비교한 결과다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data100**  **Epochs50**  **Size(128)** | **VGG16**  **Flatten** | **VGG16**  **Flatten**  **dropout** | **VGG16**  **Flatten**  **Dropout**  **(batchnorm)** |
| **Train acc** | **1.0** | **1.0** | **1.0** |
| **Train loss** | **0.0050** | **0.0033** | **0.0010** |
| **Train acc** | **0.8666** | **0.8333** | **0.7833** |
| **Train loss** | **0.5632** | **0.5033** | **0.7833** |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data100**  **Epochs50**  **Size(256)** | **VGG16**  **Flatten** | **VGG16**  **Flatten**  **dropout** | **VGG16**  **Flatten**  **Dropout**  **(batchnorm)** |  |
| **Train acc** | **0.9911** | **1.0** | **1.0** |  |
| **Train loss** | **0.0018** | **0.0019** | **0.0010** |  |
| **Train acc** | **0.9166** | **0.85** | **0.7833** |  |
| **Train loss** | **0.3487** | **0.7097** | **0.9573** |  |



input\_size를 128,128 에서 256,256으로 늘렸을 때 좋은 결과를 확인할 수 있었고, Loss 그래프를 보면 input\_size 256,256이 더 좋은 결과를 얻을 수 있다.

현재 나온 결과 중 제일 좋았던 결과는 데이터의 전체 개수는 100개 input\_size는 256,256,

모델은 flatten->dense(256)->relu->dense(3)->softmax 설정했을 때 결과가 가장 좋았다.

**Ⅳ. CONCLUSION**

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷, 음식, 사람들, 레스토랑이(가) 표시된 사진

자동 생성된 설명스크린샷, 물, 전면, 앉아있는이(가) 표시된 사진

자동 생성된 설명

다음 사진은 지금까지 제일 좋았던 모델을 적용한 결과다

‘성’ 이라는 한정된 사진을 학습시켜서 높은 확률로 결과를 얻을 수 있었다.

이번 실험에서 아쉬웠던 점은 데이터수집이 미흡했고, 모델의 capacity를 데이터의 개수에 따른 변경을 했어야 했는데, 최적의 지점을 찾는데 시간이 많이 소요되어 난항을 겪었으며, 데이터의 개수를 100~200장 정도로 적은 표본을 가진 채 실험을 진행하다 보니 실험 결과의 신빙성이 떨어지고, Model의 capacity를 찾는다면 더 좋은 결과가 나올 수 있었다고 생각한다. 이러한 문제점을 기반으로 하여 기회가 된다면 좀 더 보완하여 만들 수 있을 것이다.

**REFERENCES**

참고자료 :