석사학위 논문

딥러닝을 이용한 어종 판별 시스템

Fish Species Recognition using Deep Learning

2019년 6월

숭실대학교 정보과학대학원

소프트웨어공학과

강 혜 진

석사학위 논문

딥러닝을 이용한 어종 판별 시스템

Fish Species Recognition using Deep Learning

2019년 6월

숭실대학교 정보과학대학원

소프트웨어공학과

강 혜 진

석사학위 논문

딥러닝을 이용한 어종 판별 시스템

지도교수 최 형 일

이 논문을 석사학위 논문으로 제출함

2019년 6월

숭실대학교 정보과학대학원

소프트웨어공학과

강 혜 진

강 혜 진 의 석 사 학 위 논 문 을 인 준 함

심 사 위 원 장 이 원 철 인

심 사 위 원 정 재 만 인

심 사 위 원 최 형 일 인

2019년 6월

숭실대학교 정보과학대학원

목 차

표 목 차

그 림 목 차

국문초록

딥러닝을 이용한 어종 판별 시스템

강 혜 진

소프트웨어공학과

숭실대학교 정보과학대학원

최근 미디어와 SNS의 영향으로 한국 낚시 시장이 급격히 성장하면서 해양 레저 스포츠에 관심이 높아지고 있는 반면, 수산자원의 번식 및 효율적인 관리를 위한 수자원의 포획/채취 금지 및 체장을 규정한 수산자원관리법 시행령이 시행되고 있는 점은 대부분 잘 알지 못한다. 또한 어종을 막론하고 크기가 자신보다 작은 물고기를 먹어 치워 국내 생태계를 위협하고 있는 외래종에 대한 문제 등 어종에 대한 판별이 잘 되지 않는 일반인들이 문제를 받아들이기 쉽지 않다.

본 논문에서는 이러한 문제를 해결하기 위해 사전학습모델을 재학습한 CNN(Convolution Neural Network) 기반의 전이학습 모델이 배포되어 있는 서버를 구축하고, 사용자가 어종 이미지를 스마트폰을 통해 업로드하면 서버에서 어종을 판별하여 결과 값을 다시 전송해주는 시스템으로 누구나 스마트폰을 통해 손쉽게 어종 판별을 할 수 있는 시스템을 제안한다.

ABSTRACT

Fish Species Recognition using Deep Learning

Kang, Hye Jin
Department of Software Engineering
Graduate School of Information Science
Soongsil University

Recently, as the Korean fishing market has grown rapidly due to the influence of media and SNS, interest in marine leisure sports is increasing.

However, most of them do not know that the Enforcement Ordinance of Fisheries Resources Management Act, which stipulates prohibition of catching / collecting of water resources for the breeding and efficient management of fishery resources, In addition, regardless of the type of fish, it is not easy for general people to eat fish smaller than their own size and to identify fish species such as alien species that threaten domestic ecosystems.

In this paper, to solve these problems, we build a server on which a CNN (Convolution Neural Network) based transfer learning model

that re-learns the pre-trained model, and when a user uploads the image of a fish species through a smartphone, And sends the result back to the system. This system can easily identify fish species through smartphone.

제 1 장 서 론

1.1 연구 배경 및 목적

최근 '도시어부', '빅피쉬' 등의 낚시를 주제로 한 방송 및 예능프로그램이 많이 생기면서 낚시가 인기 여가 활동으로 떠올랐지만, 낚시터나 낚시 어선 업 등의 서비스업은 요구수준에 비해 사업의 규모가 영세하여 고품질의 서비스 제공에 한계가 있다. 또한 낚시터 환경 및 수산자원에 대한 부작용이 일반인들에게 많이 인식되지 못해 이에 대한 대처가 많이 미흡한 상태로 낚시 추, 떡밥 등으로 인한 환경오염, 낚시 대상 어족자원 감소 위험, 낚시인 안전사고 등의 문제가 커지고 있다. 특히 낚시 어획량은 내수면, 연근해 어획량의 약 18%를 차지할 정도로 수산업에 미치는 영향이 크다[5].

이처럼 낚시 인구는 증가하고 있지만, 수산자원의 번식 및 효율적인 관리를 위한 수자원의 포획, 채취 금지 및 체장을 규정한 「수산자원관리법 시행령」이 시행되고 있는 점은 일반적으로 관심을 가진 사람이 아니면 잘 알지 못하는 규정이 시행되고 있다.

또한 '큰입 배스'와 '블루길'과 같이 엄청난 식욕으로 국내 생태계에서 최상위에 자리 잡으며 작은 곤충부터 작은 물고기, 치어, 알까지 모두 먹어치우며 국내 생태계를 위협하고 있는 외래종에 대해 환경부는 2007년부터 생태계 교란 외래 생물 종에 대한 모니터링을 진행하고 있으며, 2011년부터 외래종 16종에 대하여 매년 조사를 진행하고 있다[4].

이렇듯 국내 생태계를 위협하고 있는 외래종에 대한 정보 및 채집하였을 경우 처리방법, 어종 별로 적용되는 금지 시기 및 체장에 대한 규정이 모두 달라 어종에 대한 판별조차 잘 하지 못하는 일반인들이 해당 시행령을 받아들이기에 어려운 실정이다. 예를 들어 참돔의 경우는 금지되

는 체장은 24cm이고, 가자미의 경우 금지 체장은 15cm에 12월1일부터 1월 31일 까지는 포획 금지 기간으로 지정되어 있다.

[표 1-1] 수산자원의 포획/채취 금지 체장 또는 체중(일부)

어류	포획채취금지 체장
개서대	26cm 이하
문치가자미	15cm 이하
참가자미	12cm 이하
감성돔	20cm 이하
돌돔	24cm 이하
참돔	24cm 이하
황돔	15cm 이하
넙치	21cm 이하
농어	30cm 이하
대구	30cm 이하
도루묵	11cm 이하
명태	27cm 이하
민어	33cm 이하
방어	30cm 이하
볼락	15cm 이하
붕장어	35cm 이하
조피볼락	23cm 이하

건전한 레저낚시 문화와 수산자원의 보호를 위해 우선적으로 어종에 대한 판별을 스마트폰을 이용해 손쉽게 할 수 있는 방법을 제안한다.

1.2 연구 내용

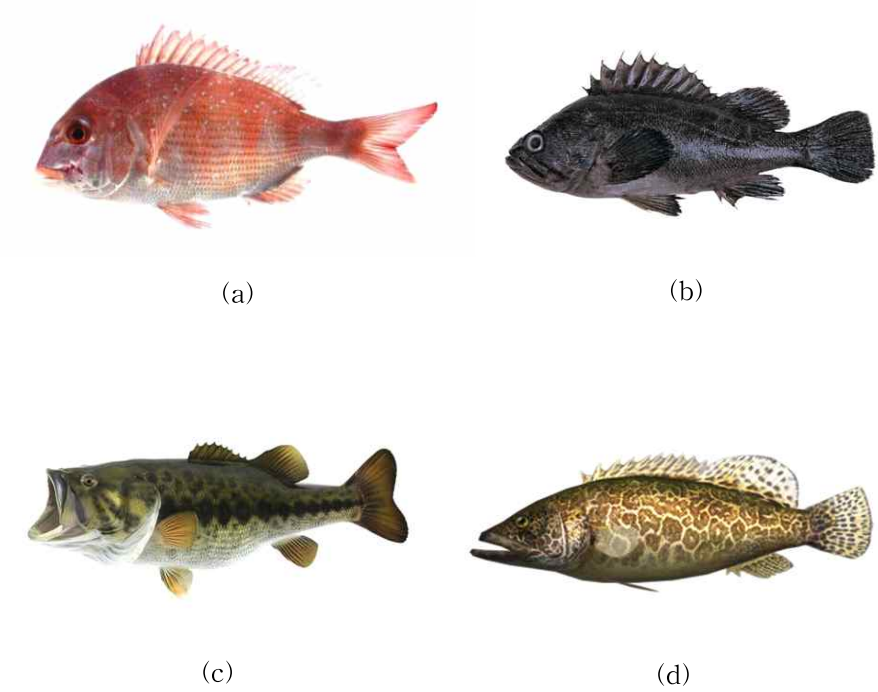
본 논문에서는 우리나라 주변에서 많이 서식하는 어종 중 낚시에 많이 이용되고 있는 4개의 어종을 대상으로 각 어종 당 최소 300장 이상의 데이터를 딥러닝을 통해 학습하며, 판별하고자 하는 물고기의 이미지를 클라우드 서버로 전송하여 학습된 내용을 바탕으로 이미지의 어종을 분류 후

그 결과 값을 다시 전송한다.

학습 대상 어종은 바다 어종 2종, 민물 어종 2종으로 각 어종별 특징은 다음과 같다.

- [바다] 참돔 : 우리나라 전 연근해, 발해만, 동중국해, 남중국해, 대만 근해에 분포하고 있으며 몸의 형태는 타원형으로 외곽이 올라가 있으며, 측편 한다. 양 턱의 옆쪽으로 2줄의 큰 어금니가 줄지어 있으며, 위으 앞쪽에는 2쌍, 아래턱 앞쪽에는 3쌍의 송곳니가 있다. 등지느러미 가시는 강하고 뺨에는 6~8줄의 비늘이 있다. 꼬리지느러미 끝 가장자리는 검은 색을 띤다. 몸 빛깔은 일반작으로 담홍색을 띠고 등 쪽에는 청록색의 작은 반점들이 많이 흩어져 있으며 [그림 1-1] (a)와 같다[8].
- [바다] 우럭(조피볼락) : 우리나라 전 연안, 일본 북해도 이남, 중국 북부 연안에 분포하고 있으며 몸은 긴 타원형으로 측편 되어 있으며 꼬리지느러미 뒤끝은 수직 형이거나 약간 볼록하다. 아래턱이 윗 턱보다 약간 돌출되어 있으며, 양턱에는 융털모양의 이빨 띠가 있다. 몸 빛깔은 짙은 회갈색으로 눈 뒤쪽으로 향한 비스듬한 2~3줄의 흑색 띠가 있으며 [그림 1-1] (b)와 같다[9].
- [민물] 배스 : 물의 흐름이 없는 정수역인 호수나 유속인 느린 하천 하류에서 서식한다. 농어목 검정우럭과에 속하는 민물 어류의 일종으로 북아메리카 대륙이 원산지이며 식용, 레저의 목적으로 여러 나라에 도입되었다. 국내에도 식용으로 수입되었으나 현재는 지나친 개체 수 증가로 인해 생태계교란 생물로 지정이 되었다. 등 부분은 진한 올리브색, 배쪽은 흰색에 가까운 담녹색이며 몸통 가운데를 지나는 어두운 색의 불규칙한 반점으로 이루어진 세로줄이 있으며 [그림 1-1] (c)와 같다[11].

- [민물] 쏘가리 : 압록강, 서해와 남쪽 연해에 있는 여러 하천 중상류에 분포한다. 몸 길이는 50cm이상 자라며, 몸은 긴 편이며 측면은 옆으로 납작하다. 몸은 작은 둥근 비늘로 덮여 있으며, 아래턱이 윗 턱보다 길다. 지느러미에는 가시가 있고 꼬리지느러미는 갈라지지 않았고 둥근 편이다. 몸 빛깔은 전체적으로 황갈색이며, 몸 표면에 짙은 흑갈색 반점이 있고 가슴지느러미를 제외한 나머지 지느러미에도 반점이 있으며 [그림 1-1] (d)와 같다[10].



(a) (b)

(c) (d)

[그림 1-1] 학습대상 어종 이미지 :
(a) 참돔, (b) 우럭, (c) 배스, (d) 쏘가리

1.3 논문 구성

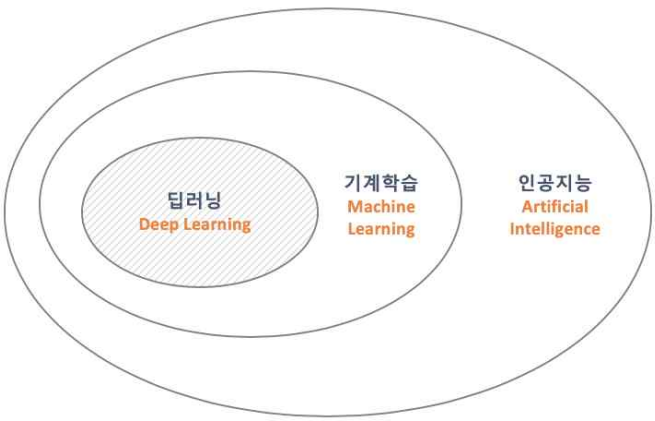
본 논문의 구성은 다음과 같다. 2장에서는 딥러닝과 신경망 모델에 대한 관련 연구에 대해 설명한다. 3장에서는 어종 판별을 위해 제안한 CNN을 활용하여 학습방법 및 시스템 구성을 설명한다. 4장에서는 전이 학습을 통한 모델별로 동일한 학습 데이터를 학습 후 비교 실험 및 분석을 하고 5장에서는 본 논문의 전체적인 결론을 맺는다.

제 2 장 딥러닝과 신경망 모델에 대한 관련 연구

2.1 딥러닝

딥러닝은 다량의 데이터나 자료들의 훈련 데이터를 통해 핵심적인 내용 또는 기능을 예측 및 분류하는 기계학습의 한 분야로, 특정 데이터가 있으면 이를 컴퓨터가 알아 들을 수 있는 형태로 표현 및 학습에 적용하기 위한 많은 연구가 진행되고 있으며, 다양하고 많은 딥러닝 기법들이 음성인식, 자연어 처리, 컴퓨터 비젼, 음성/신호 처리 등의 분야에 적용되어 최첨단의 결과들을 보여주고 있다[7].

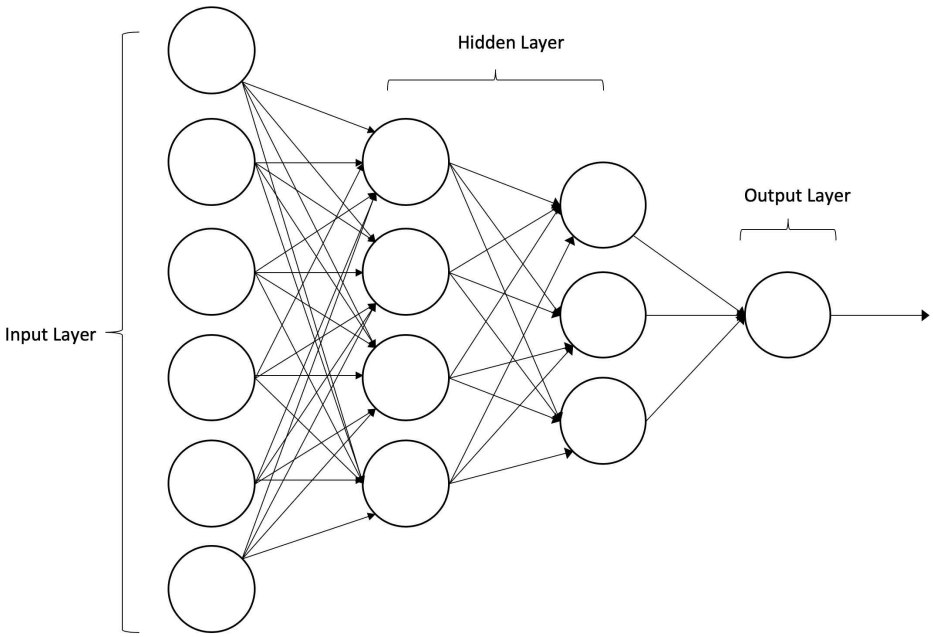
기계학습과는 유사하지만 인간의 신경망을 모델화 하여 새로운 데이터 셋을 예측하는 기술로 이미지나 음향/음성 및 동영상 데이터의 패턴 분석 학습을 통해 스스로 무엇인지를 인지하는 기술이다.



[그림 2-1] 인공지능과 기계학습, 딥러닝 범주

2.1.1 심층 신경망 (Deep Neural Network, DNN)

전통적인 기계학습 알고리즘은 하나의 입력층(Input Layer)와 하나의 출력층(Output Layer)로 이루어져 있으며 사이에 하나의 은닉층을 가지고 있는 반면, 심층 신경망은 입력층인 Input Layer와 츌력층인 Output Layer 사이에 여러 개의 은닉층(Hidden Layer)들로 이루어진 인공신경망(Artificial Neural Network, ANN)이다.



[그림 2-2] 심층 신경망 구조

2.1.2 심층 신경망의 문제점

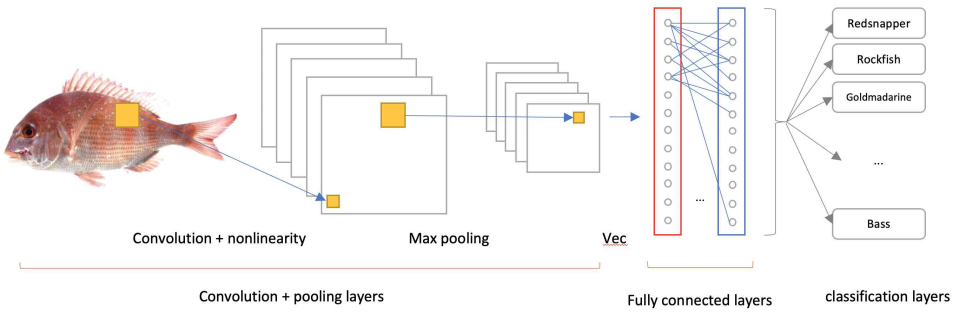
기존의 ANN과 같이, 심층 신경망 역시 Naive한 방식으로 학습될 경우 많은 문제가 발행한다. 대표적으로 과적합과 높은 시간 복잡도를 볼 수 있다.

과적합 문제는 추가된 계층들이 학습 데이터의 rare dependency의 모형화가 가능하도록 해주기 때문으로, 과적합을 극복하기 위해서는 weight decay(l2-regularization) 또는 sparsity(l1-regularization)와 같은 regularization방법을 사용할 수 있다[7]. 최근에 많이 사용되고 있는 정규화 방법 중에서 드롭 아웃(drop out) 정규화가 있다.
dropout 정규화는 학습 도중 은닉계층들의 몇몇 유닛들이 임의로 생략되어 학습 데이터에서 발생할 수 있는 rare dependency를 해결한다.

오차역전파법과 경사 하강법은 구현이 편리하고 국지적 최적화(Local Optima)에 잘 도달한다는 특성으로 선호된 방법이지만 심층 신경망 학습 시 시간 복잡도가 매우 높은 문제가 있다. 심층 신경망 학습 시 계층의 수와 계층 당 유닛 수, 학습률, 초기 가중치 등등 많은 매개별수들이 고려되어야 한다. 이러한 시간 복잡도를 해결하기 위해 미니 배치(mini batch, 여러 학습 예제들의 경사를 동시에 계산), 드롭 아웃(drop out)과 같은 방법을 통해 해결한다. 또한 행렬 및 벡터 계산에 특화된 GPU는 많은 처리량을 바탕으로 두드러지는 학습 속도 향상을 보여준다[7].

2.2 CNN (Convolution Neural Network)

CNN은 이미지 및 영상 데이터 처리에 특화된 심층 신경망 중 하나로 입력된 영상에서 특징을 추출하고, 추출된 특징을 이용해 해당 영상에 있는 물체가 어떤 물체인지 분류한다.



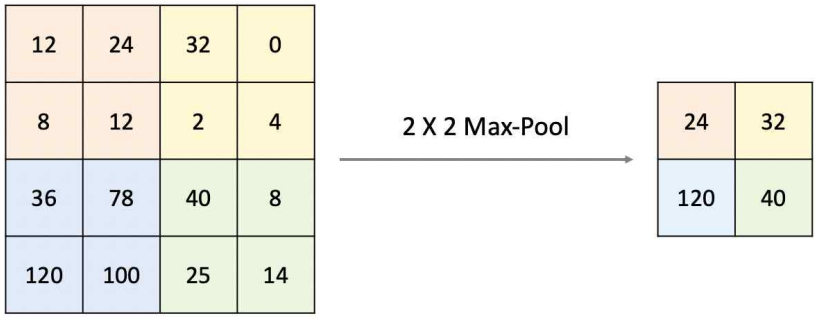
[그림 2-3] CNN 레이어 구조

- 합성곱 계층(Convolution Layer)

주로 사용될 이미지 데이터는 width, height, channel 3차원으로 구성된 데이터로 합성곱 계층에서 입력데이터로 이미지의 3차원 데이터를 입력하고 3차원의 데이터로 출력하므로 형상을 유지할 수 있다. 이러한 입출력 데이터를 특징맵(Feature Map) 이라고 한다. 합성곱의 연산은 필터(커널)의 윈도우를 일정한 간격으로 이동해가며 계산하여 입력데이터와 필터간의 서로 대응하는 항목끼리 곱한 후 총합을 구하게 된다.

- 풀링 계층(Pooling Layer)

이미지의 종류를 판별할 때 중요한 것은 필터를 거친 출력 결과를 그대로 분류에 사용하는 것이 아니라 이미지의 해상도를 낮춰 상세 정보를 제거하게 되는데 이와 같은 처리를 하는 것이 풀링 계층이다. 이러한 풀링 계층에서 사용되는 것은 다양한 방법이 있으나 가장 많이 사용되는 것은 맥스 풀링(max pooling)으로 정해진 영역 안에서 제일 큰 값만 다음 층으로 전송하고 나머지는 버리는 방법이다. [그림 2-4]와 같이 원본 이미지를 4개의 구역으로 나누어서 2 X 2의 max pooling을 적용하면 각각의 영역에서 가장 큰 값인 24,32,120,40을 가지고 새로운 층을 만들어 낼 수 있다.



[그림 2-4] Max Pool 설명 예시

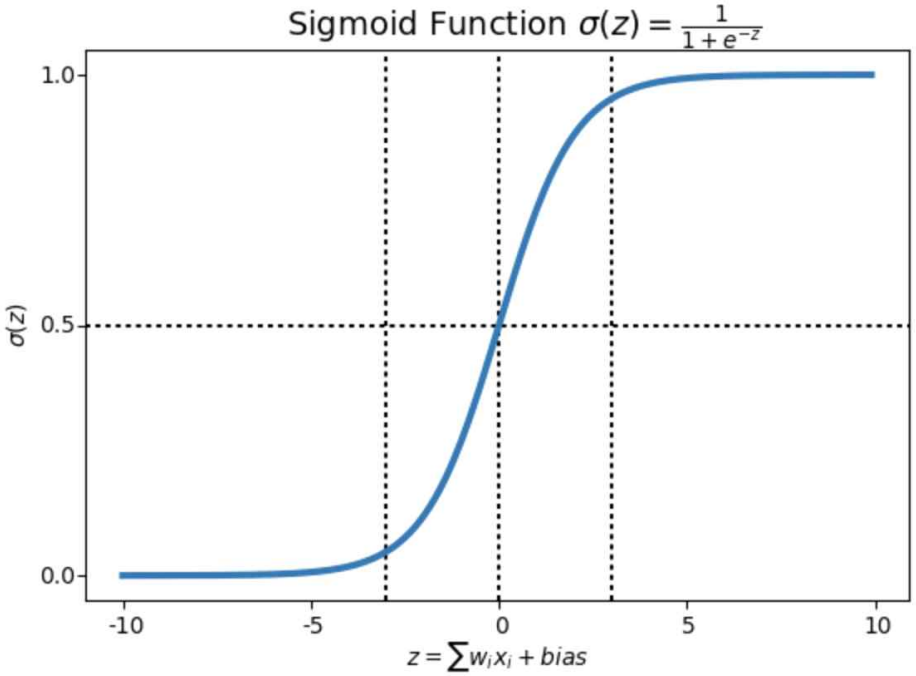
- 완전 연결 계층 (Fully Connected Layer)

Convolution Layer에서 특징 추출이 완료되면 추출된 특징 값을 기존의 인공 신경 지능망에 넣어 분류를 한다. 입력데이터와 가중치가 1대 1로 대응하는 것을 완전 연결(Fully Connected)라고 한다.

- 활성 함수 (Activation Function)

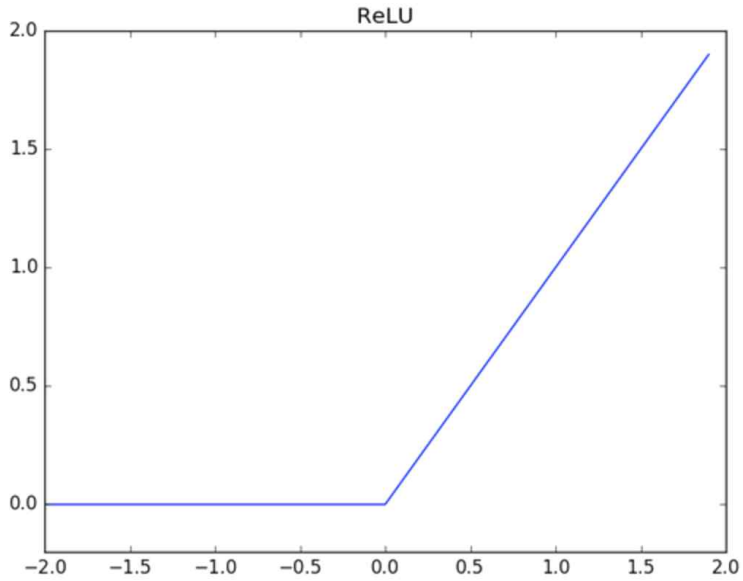
필터를 통해 특징맵을 추출한 다음 활성함수를 적용한다. 활성함수란 학습 데이터에서 특징이 있는지 없는지에 대해 정량적이 값을 비선형 값으로 바꾸는 역할을 하며, 시그모이드(sigmoid), ReLU 함수 등이 있다.

시그모이드 함수는 결과 값을 0,1로 나타내는 것이 아니라 0~0.5 에 가까우면 0을, 0.5 ~ 1 사이에 가까우면 1을 출력한다.



[그림 2-5] 시그모이드 함수 그래프

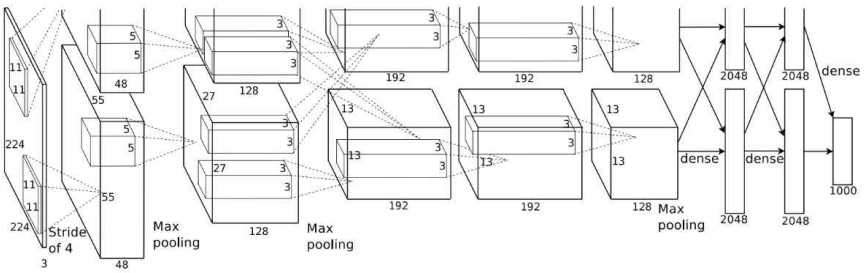
ReLU 함수는 입력이 0이상이면 그 입력 값을 그대로 출력하고, 그 이하인 경우에는 0을 출력한다. CNN에서 시그모이드 함수보다 ReLU함수를 더 많이 사용하는데 그 이유는 신경망의 층이 깊어질수록 학습이 어려워지므로 전체의 레이어를 한번 계산 한 후 그 값을 재활용하는 Back Propagation 이라는 방법을 사용하는데, 시그모이드 함수를 활성함수로 사용할 경우 Gradient Vanishing 현상 때문에 제대로 작동하지 않기 때문이다.



[그림 2-6] ReLU 함수 그래프

CNN의 주요 모델의 종류와 특징은 다음과 같다.

2.2.1 AlexNet



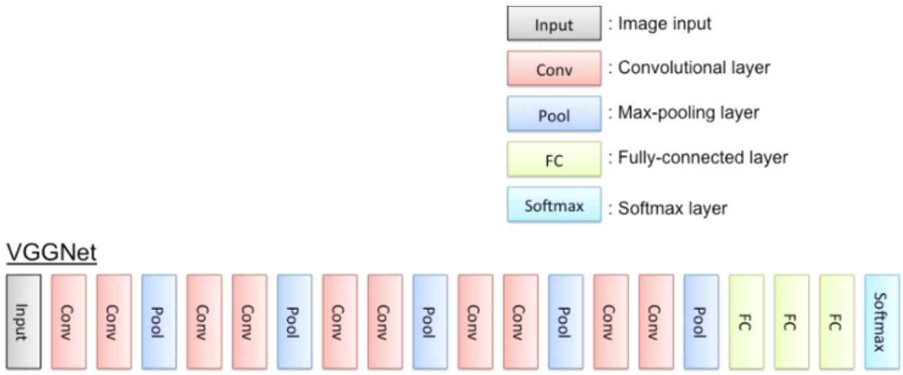
[그림 2-7] AlexNet 구조도

AlexNet은 2012년 개최된 ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 대회의 우승을 차지한 컨볼루션 신경망 구조이

다. 2개의 GPU로 병렬연산을 수행하기 위해 병렬적인 구조로 설계된 점이 가장 큰 특징이다[2]. [그림 2-7]에서 보면 AlexNet 은 conv layer, max-pooling layer, dropout layer 5개와 fully connected layer 3개를 가지고 있다.

2.2.2 VGGNet

AlexNet 이후 층을 더 깊게 구성하여 성능을 높인 네트워크 모델들이 등장하였는데 그 중 하나가 VGGNet이다. VGGNet은 간단한 구조와 단일 네트워크에서 좋은 성능을 보여주는 모델로 Convolution layer는 kernel 사이즈 3x3, padding 1로 설정하여 이미지를 resize 하는 것이 아니라 max pooling을 사용하여 이미지를 resize 한다.

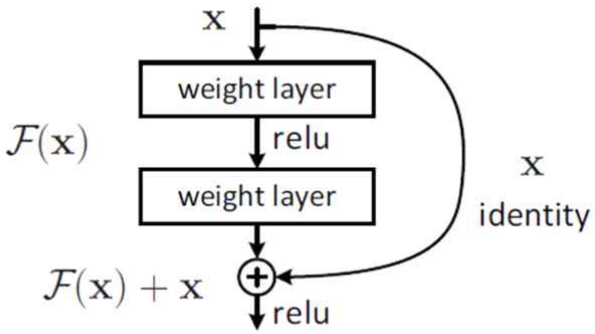


[그림 2-8] VGGNet 구조도

2.2.3 ResNet

2015년 ILSCVRC에서 오류율 3.6%로 1등을 차지한 모델로 인간의 분류 오차가 5~10%인걸 감안하면 놀라운 성능을 보인다. 층이 깊어질수록 역전파되는 그래디언트가 전달되지 않아 학습이 잘 되지 않는 Gradient

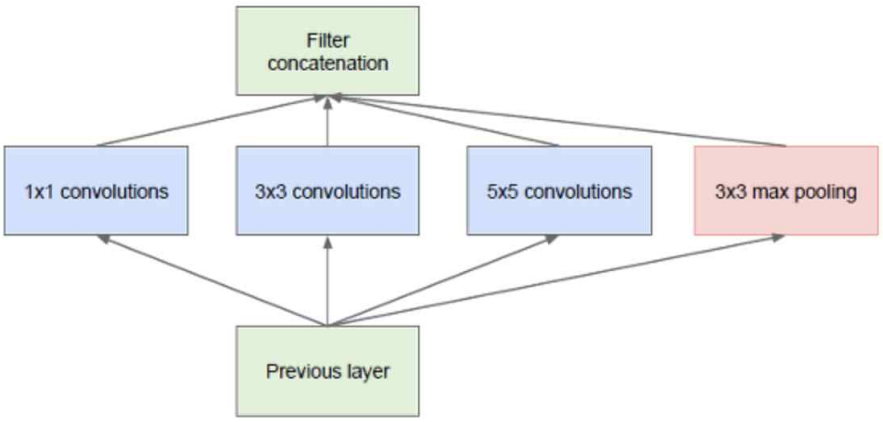
Vanishing 문제를 ResNet에서는 그래디언트가 잘 전달될 수 있도록 Residual Learning 블록을 사용한다. [그림 2-9] 과 같이 입력에서 바로 출력으로 연결되는 shortcut연결이 생기고, 파라미터가 없이 연결되는 구조이므로 연산량의 관점에서는 덧셈이 추가되는 것 외에는 차이가 없다. 이전에는 H(x)를 구하기 위해 학습했다면 F(x)가 거의 0 이 되는 쪽으로 학습하면 입력의 작은 움직임을 쉽게 검출할 수 있게 되고 나머지를 학습한다는 관점에서 residual learning이라고 한다.



[그림 2-9] Residual Learning 블록

2.2.4 InceptionV3

2014년 IRSVRC에서 1등한 모델로 대회에서는 GoogLeNet이라고 불렀으며, 대회 이후 Going Deeper with Convolutions 라는 논문에서 Inception이란 이름으로 발표했다. InceptionV3는 GoogLeNet을 발전시킨 형태로 뉴럴넷 안에서 차원을 과도하게 줄였을 때 정보의 양이 크게 줄어드는 현상(Representational Bottleneck)에 대한 해결책으로 특징맵의 크기를 효과적으로 줄이는 방법으로 병렬적으로 수행하고 합치는 방법을 사용하였으며, Inception module은 [그림 2-10]과 같다[6].

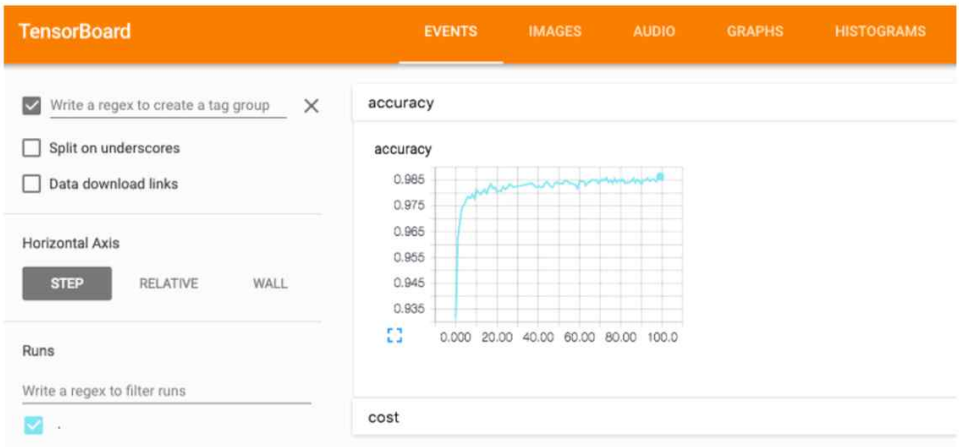


[그림 2-10] Inception module

2.3 Tensorflow와 Keras

텐서플로(Tensorflow)는 구글이 공개한 머신러닝 라이브러리로 오픈소스로 공개되어 있다. 특히, 딥러닝에 적용할 것을 고려하여 설계되어 있으며 다양한 사례와 예제 코드가 함께 소개되고 있다.

본 논문에서도 실험을 위한 주 라이브러리로 텐서플로와 케라스를 이용하였으며, 학습을 통해 얻은 로그 데이터를 그래프 형태로 시각화 시켜주는 도구인 텐서플로를 통해 실험 결과를 보여준다.



[그림 2-11] 텐서보드 화면

케라스(Keras)는 텐서플로와 통합되어 있는 고수준 API로 레고 블록을 조립하는 듯한 간단한 모듈 구성이 특징이 있다.

제 3 장 제안한 어종 판별 방법

3.1 이미지 수집 및 전처리

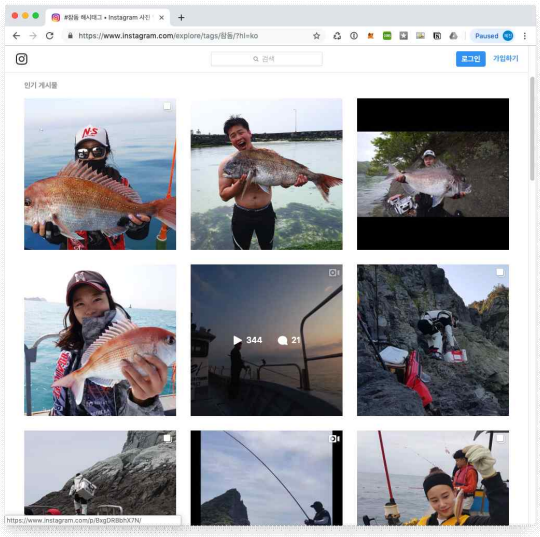
본 논문에서 사용할 어종은 총 4종으로 각 어종별 최소 300장 이상의 이미지를 기준으로 학습한다. 학습을 위한 이미지 수집을 위해 인스타그램 등에서 검출된 페이지를 크롤링하여 수집하기 위한 웹 크롤러를 개발하여 이용하였으며, 이렇게 수집된 이미지들은 종류별로 구분하여 라벨링을 하여 최종 학습데이터로 준비한다.

3.1.1 이미지 수집 방법

이미지 수집 대상은 인스타그램 및 네이버, 구글 등으로 지정하였으며, 인스타그램을 통한 수집을 위한 검색어는 학습하고자 하는 4종의 어종 명을 해시태그로 지정한 포스트에 대해 결과를 가져올 수 있게 하였으며 그 URL은 [표 3-1]과 같다. 또한 해당 URL을 브라우저를 통해 실행시켜보면 [그림 3-1]과 같은 결과화면을 볼 수 있다.

[표 3-1] 해시태그 검색을 통한 웹 크롤링 대상 URL

Tag	URL
참돔	https://www.instagram.com/explore/tags/%EC%B0%B8%EB%8F%94/?hl=ko
우럭	https://www.instagram.com/explore/tags/%EC%9A%B0%EB%9F%AD/?hl=ko
배스	https://www.instagram.com/explore/tags/%EB%B0%B0%EC%8A%A4/?hl=ko
쏘가리낚시	https://www.instagram.com/explore/tags/%EC%8F%98%EA%B0%80%EB%A6%AC%EB%82%9A%EC%8B%9C/?hl=ko



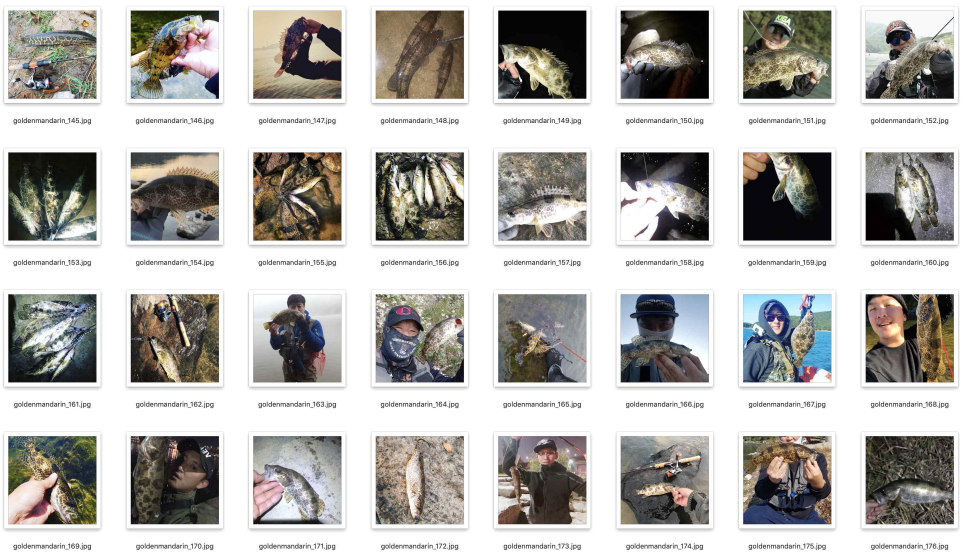
[그림 3-1] 검색 결과 화면

웹 크롤러 개발 언어는 Python을 사용하였으며, 웹 사이트에 접근하여 자동화 테스트를 위해 Selenium 라이브러리와 다운받은 html코드를 객체 구조로 변환하기 위해 파싱해주는 BeautifulSoup 라이브러리를 이용하였다. 웹 크롤링을 통해 수집된 이미지들은 학습의 정확도를 위해 일일이 눈으로 확인하여 조건에 맞지 않는 이미지는 제외하였으며 그 조건은 다음과 같다.

- 이미지에 해당 어종이 포함되어 있을 것
- 하나의 이미지에는 하나의 어종만 있을 것
- 육안으로 어종에 대한 구별이 가능한 이미지

[표 3-2] 최종 학습데이터 갯수

Classes	Train	Validation	Test	Total
참돔	300	100	100	500
우럭	300	69	100	469
배스	300	100	100	500
쏘가리	300	100	100	500



[그림 3-2] 이미지 수집한 결과 일부 (쏘가리 수집)

3.1.2 전처리 방법

준비된 학습 데이터 셋의 개수가 어종별 500장에서 validation 용 100장과 테스트용 100장을 제외한 학습용 데이터가 300장 정도로 작기 때문에 조금 더 정확한 학습을 위해 Keras에서 제공하는 ImageDataGenerator를 이용하여 학습 데이터를 늘리도록 하였다. ImageDataGenerator에서 제공하는 옵션은 다음과 같다.

- rotation_range : 지정된 각도 범위 내에서 임의로 원본이미지를 회전시킨다.

- width_shift_range : 지정된 수평 방향의 이동 범위 내에서 임의로 원본 이미지를 좌우로 이동시킨다.
- height_shift_range : 지정된 수직 방향의 이동 범위 내에서 임의로 원본 이미지를 상하로 이동시킨다.
- shear_range : 밀림 강도 범위 내에서 임의로 원본 이미지를 변형시킨다. 수치는 시계반대방향으로 밀린 강도를 라디안으로 나타낸다.
- zoom_range : 지정된 확대/축소 범위 내에서 임의로 원본이미지를 확대/축소한다.
- horizontal_flip : 수평 방향으로 뒤집기를 한다.
- vertical_flip : 수직 방향으로 뒤집기를 한다.

3.2 CNN을 이용한 학습

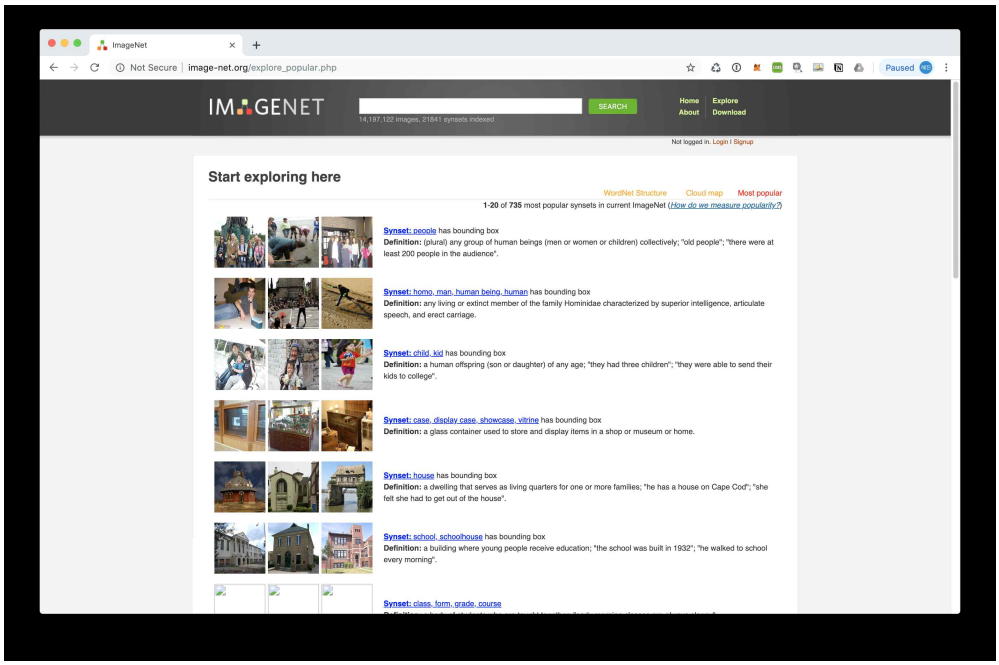
본 논문에서는 특징을 직접 학습하므로 수동으로 특징을 추출할 필요가 없으며 이미지 분류에 제일 많이 사용되고 있는 CNN을 이용하여 학습을 진행하였다. 이미지 수집 단계에서 수집된 이미지를 640 x 640 사이즈로 일괄 조정 하였고, 실제 학습 시에는 각 모델별 입력 값에 맞는 사이즈로 resize 하여 사용한다.

3.2.1 전이 학습

딥러닝 모델을 처음부터 구축하여 학습하려면 고성능 GPU가 필요하고 네트워크의 규모가 크면 학습 시 오랜 계산 시간과 리소스가 필요하다. 이러한 문제를 해결하기 위해 사전학습 모델을 재학습 하는 전이 학습을 이용한다.

사전 학습 모델(Pre-Trained Model)이란 사전에 어떤 주제에서 가중치가 학습되어 있는 딥러닝 모델로 최첨단 네트워크 구조를 이용하고 있어

높은 정확도를 기대 할 수 있다. 또한 대규모의 이미지 데이터 세트를 이용해서 학습을 하였기 때문에 일반적인 이미지 분류 문제에 사용할 때는 정확도나 필요한 분류 클래스도 충분히 준비되어 있다. 사전 학습 모델의 학습 데이터로 널리 사용되고 있는 이미지 데이터의 하나로 ImageNet이 있다. ImageNet은 연구 목적으로 수집된 대규모의 임지 데이터 세트로 동물과 식물, 탈것과 같은 대표적인 클래스 분류와 이미지가 포함되어 있다.



[그림 3-3] ImageNet 사이트 화면

ImageNet의 데이터 세트를 이용하여 성능을 비교하는 이미지 인식 대회인 ILSVRC(ImageNet Large Scale Visual Recognition Challenge)가 2017년 까지 개최되었으며, 주요 모델의 종류 및 특징은 [표 3-3]과 같다.

[표 3-3] 사전학습 모델의 종류 및 특징

Network	Depth	Size (MB)	Input Size	Parameters (Millions)
alexnet	8	227	227x227	61
vgg16	16	515	224x224	138
vgg19	19	535	224x224	144
squeezenet	18	4.6	227x227	1.24
googlenet	22	27	224x224	7
inceptionv3	48	89	299x299	23.9
densenet201	201	77	224x224	20
mobilenetv2	53	13	224x224	3.5
resnet18	18	44	224x224	11.7
resnet50	50	96	224x224	25.6
resnet101	101	167	224x224	44.6
xception	71	85	299x299	22.9
inceptionresnetv2	164	209	299x299	55.9
shufflenet	50	6.3	224x224	1.4
nasnetmobile	*	20	224x224	5.3
nasnetlarge	*	360	331x331	88.9

본 논문에서 분류하고자 하는 어종에 대한 분류 대상의 클래스가 ImageNet에 존재하지 않기 때문에 어종 이미지를 사전 학습 모델을 이용하여 재학습 시키는 전이 학습(Transfer Learning)을 이용한다.

전이학습을 이용할 경우 네트워크 구조의 대부분을 그대로 사용하기 때문에 네트워크를 정의할 필요가 없으며, VGG16과 같이 이미 많은 수의 클래스를 인식할 수 있는 특징량 추출기를 가진 모델은 어종 자체를 학습하지 않아도 어종 별 차이를 구별하는데 유효한 특징을 이미 학습하고 있을 가능성도 있기 때문에 처음부터 가중치를 학습시키는 것보다 적은 데이터와 시간으로 좋은 정확도를 낼 수 있다.

아래 [그림 3-4]과 [그림 3-5]는 사전학습모델인 VGG16의 기본 레이어

구조와 본 논문에서 학습을 위해 변경한 레이어 구조에 대한 설명이다.

Layer (type)	Output Shape	Param
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPolling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Total Params	14,714,688
Trainable Params	14,714,688
None-trainable Params	0

[그림 3-4] 사전학습모델 - VGG16 레이어구조

학습을 위해 사용된 사전학습 모델에서 재학습이 필요하지 않은 층에 대해서는 학습 불가 설정을 하고, 새로운 층을 추가하였다. 합성곱 층의 출력을 완전연결 계층에 추가하기 위한 Flatten층을 추가하고 가중치 학습을 위한 완전연결 계층인 Dense 레이어를 추가한 이후 오버피팅 등을 없애기 위한 Dropout층을 추가하고, 마지막에 클래스 확률을 출력하는 출력층을 추가하여 새로운 모델을 구성하였다.

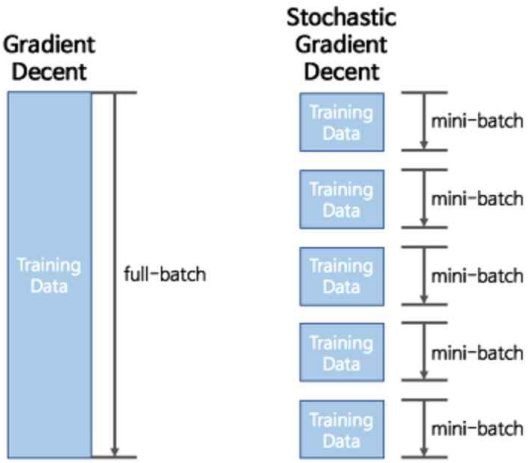
Layer (type)	Output Shape	Param
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPolling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6422784
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257

Total Params	21,137,729
Trainable Params	11,142,657
None-trainable Params	9,995,072

[그림 3-5] 재학습을 위해 수정된 레이어

3.2.2 최적화 알고리즘

뉴럴 네트워크의 weight의 기울기를 구하고 loss를 줄이는 방향으로 조정해 나가는 방법을 통해 학습을 진행하는데 기존의 gradient descent방식에는 최적 값을 찾아 나가기 위해 한 단계마다 나아갈 때 모든 데이터 셋을 설정해주어야 한다는 문제가 발생한다. 이러한 문제 때문에 나오게된 Optimizer가 SGD(Stochastic Gradient Descent) 이다.



[그림 3-6] Gradient Descent와 SGD비교

[그림 3-6]에서 설명하는 것처럼 SGD는 Mini-batch사이즈만큼 조금씩 돌려서 최적의 값을 찾는다. SGD 에도 미니 배치를 통해 최적의 값을 찾는 방향이 뒤죽박죽이고, learning rate 에 따라 값이 작으면 학습시간이 오래 걸리고, 너무 크면 최적의 값을 찾지 못하는 문제가 있다. 이러한 SGD문제를 개선한 많은 Optimizer들 중에 Adam(Adaptive Moment Estimation)이 많이 사용되고 있다.

Adam은 RMSProp와 Momentum의 장점을 합친 최적화 알고리즘이다. learning rate를 1/2씩 exponential하게 줄여 주는 기능을 수행하고

있기 때문에 learning rate decay에 대한 고려가 필요 없다[6].

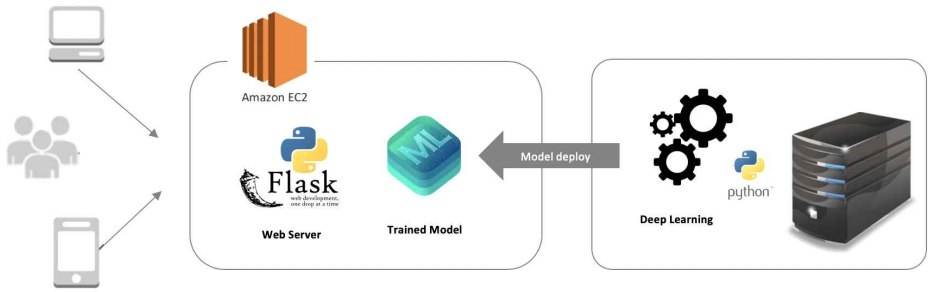
3.2.3 Training

구성된 네트워크 모델의 학습은 각 어종별 Training Data 300장과 Validation Data 약 100장, 총 1,198장의 이미지로 16batch에 30epoch으로 진행하였으며, 학습시간은 모델별로 대략 3시간 정도 소요되었다.

학습에 대한 결과 및 비교 분석은 4장을 통해 설명하기로 한다.

3.3 시스템 구성

최종 시스템 구성은 스마트폰을 통해 촬영된 이미지를 서버로 전송하고 서버에서는 학습된 모델을 통해 해당 이미지를 분류하고, 분류된 결과 값을 다시 스마트폰으로 전송해주는 시스템 구성을 제안한다.



[그림 3-7] 시스템 구성도

클라우드 서버는 AWS(Amazon Web Services)를 이용하며 서버 언어는 Python을 기반으로 한다. 클라우드 서버에 학습된 모델을 배포 한 후, API를 통해 전송된 이미지의 어종을 분류하여 예측된 결과 값을 응답 값으로 전송한다.

API 서버는 Python 기반의 Flask를 사용하여 구성하였으며, 기본적인 호출은 POST를 통해 판별할 이미지를 포함하여 호출한다.

스마트폰용 어플리케이션 개발은 크로스플랫폼 지원을 위해 Angular 기반의 Ionic Framework를 이용하여 사진촬영 및 사진선택을 통해 어종 분류 예측 값을 표시해주는 프로토타입을 개발한다.

제 4 장 실험 및 분석

이 장에서는 어종 판별을 위한 학습 시 사전 학습 모델별 성능과 최적화 알고리즘에 따른 예측 정확도로 판별 성능을 평가한다.

4.1 실험 환경

실험에 사용된 이미지는 일반적으로 스마트폰을 이용해 촬영을 하고 SNS에 업로드된 이미지로 사용하였으며, 딥러닝 학습(training)을 진행한 컴퓨터와 API를 통해 값을 전달해주는 서버의 환경은 다음과 같다.

[표 4-1] 용도별 실험 환경

구분	학습용	서버용
CPU	2.7 GHz Intel Core i7	EC2 t2.large
Graphic	Intel HD Graphics 530	
Ram	16 GB	8 GB
OS	macOS Mojave	Ubuntu
python	3.6.6	3.6.7
tensorflow	1.13.1	1.13.1

4.2 분류 모델을 통한 비교 실험

여러 종류의 사전학습 모델을 바꿔가며 학습을 시킨 결과를 비교하고자 한다. 기본적으로 사전학습 모델로 제공이 되는 네트워크 중 성능이 좋다고 알려져 있는 VGG16, InceptionV3, ResNet50 3가지 모델을 사용한다. 학습 시 동일한 비교를 위해 지원하는 1차 학습 시에는 Input 구조와 Output 층을 동일하게 변경하여 학습을 진행하였으며, 2차 학습 시에는 각 모델별로 정확도 향상을 위해 최적의 구조 및 파라미터를 찾아 학습하였다.

4.2.1 VGG16

[표 4-2] VGG16 기본 설정 값

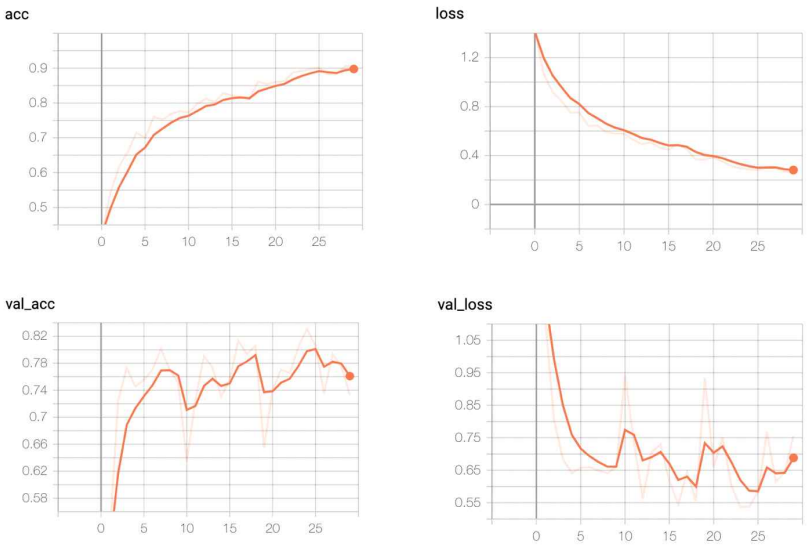
항목	1차 학습	2차 학습
loss	categorical_crossentropy	categorical_crossentropy
optimizer	SDG	SDG
learning rate	0.00004	0.0005
momentum	0.9	0.9
batch size	16	16
training time	3시간 10분	3시간 50분
accuracy	88.06%	90.29%
validation accuracy	77.80%	73.29%
test accuracy	88%	79%

VGG16모델의 1차 학습시간은 3시간 10분이 소요되었으며, 결과는 다음 표와 같이 최종 epoch에서의 accuracy는 88% 로 나왔으며 validation accuracy는 77%로 나왔다. 정확도를 조금 더 올리기 위해 learning rate를 조정하여 2차 학습을 진행하였고 그 결과는 [그림 4-1]과 같고 학습 로그는 [표 4-3]과 같다.

[표 4-3] VGG16 모델 학습 결과

epoch	acc	loss	val_acc	val_loss
0	0.41940532	1.41463428	0.47103275	1.29959491
1	0.54147105	1.06660716	0.52392947	1.11553138
2	0.61737089	0.91839905	0.72544081	0.80361394
3	0.65884194	0.84550504	0.77329975	0.68400767
4	0.71473354	0.7527737	0.74559194	0.64139553
5	0.69953052	0.75130034	0.75566751	0.65783233
6	0.76015625	0.64234201	0.77078086	0.65972263

7	0.7523511	0.64474361	0.80100756	0.64966236
8	0.76917058	0.59269023	0.77078086	0.64064932
9	0.77699531	0.57946119	0.75062972	0.6606198
10	0.77265625	0.57743553	0.63476071	0.94287233
11	0.79733959	0.53123523	0.72544081	0.73660807
12	0.81269592	0.49187395	0.79093199	0.56263243
13	0.80203443	0.5095684	0.77329975	0.7052569
14	0.8286385	0.46316251	0.73047859	0.72959436
15	0.82159624	0.44888902	0.75566751	0.61660473
16	0.81846635	0.4889444	0.81360202	0.54542314
17	0.81064163	0.45456398	0.79345088	0.64489415
18	0.86228482	0.36811577	0.80604534	0.55525467
19	0.85367762	0.36517169	0.65491184	0.93387021
20	0.86071987	0.38275271	0.74055416	0.65919031
21	0.86285266	0.35346812	0.77078086	0.75263634
22	0.88732394	0.31487957	0.76574307	0.60153769
23	0.89201878	0.29701147	0.80352645	0.53737944
24	0.89671362	0.28672488	0.83123426	0.53789985
25	0.90140845	0.28144232	0.80604534	0.58309371
26	0.88341158	0.30469623	0.73551637	0.76834656
27	0.88262911	0.30395676	0.79345088	0.614066
28	0.90610329	0.26719667	0.77581864	0.64454771
29	0.9029734	0.26981715	0.73299748	0.75722934



[그림 4-1] VGG16 학습 그래프

4.2.2 InceptionV3

[표 4-4] InceptionV3 기본 설정 값

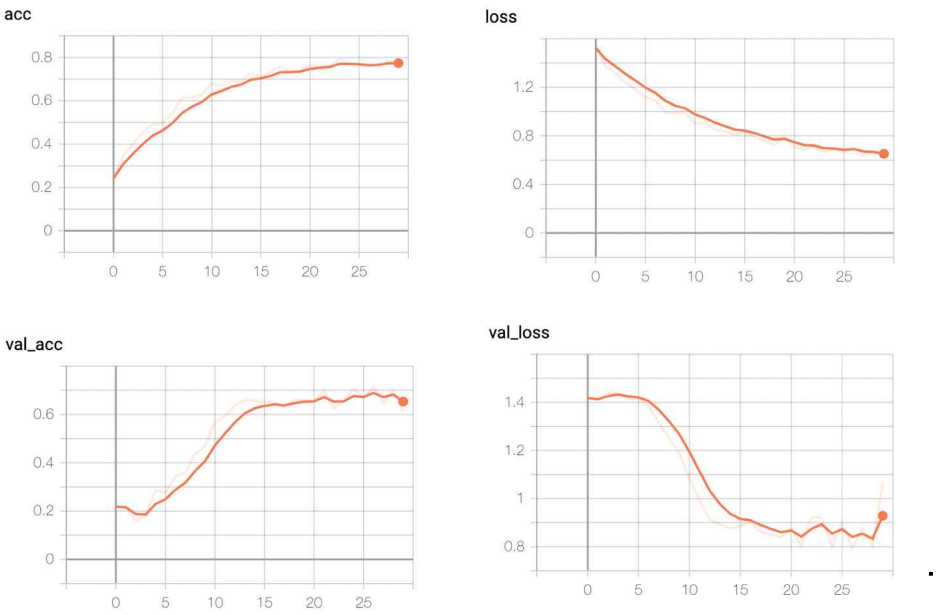
항목	1차 학습	2차 학습
loss	categorical_crossentropy	categorical_crossentropy
optimizer	SDG	SDG
learning rate	0.00004	0.0003
momentum	0.9	0.9
batch size	16	32
training time	2시간 35분	3시간
accuracy	98.99%	77.47%
validation accuracy	19.72%	60.94%
test accuracy	-	67%

InceptionV3모델의 총 학습시간은 2시간 35분이 소요되었으며, 1차 결과는 다음 표와 같이 최종 epoch에서의 accuracy는 98% 로 나왔으며 validation accuracy는 19%로 나왔다. 초기 파라미터를 통한 1차 학습에서 오버피팅등의 문제가 발생되어 4.2.1의 VGG학습법과 달리 일부 레이어를 조정하여 2차 학습을 진행하였다. 2차 학습의 결과는 accuracy 77.47%, validation accuracy 60.94%로 높지는 않지만 비교적 안정적인 학습이 되었으며 결과는 [그림 4-2]와 같고, 해당 로그는 [표4-5]와 같다.

[표 4-5] InceptionV3 모델 학습 결과

epoch	acc	loss	val_acc	val_loss
0	0.24177632	1.52418558	0.21875	1.41809131
1	0.34818482	1.37518053	0.21354167	1.40984759
2	0.40264026	1.31457501	0.16145833	1.43873714
3	0.45394737	1.23748932	0.18229167	1.44116573
4	0.49013158	1.18393823	0.28645833	1.41330232
5	0.49504951	1.1180441	0.27604167	1.41515254
6	0.54455446	1.09049896	0.34375	1.38305273
7	0.61348684	0.99422118	0.359375	1.31541224
8	0.61551155	0.98765587	0.4375	1.2517083
9	0.62664474	0.99365454	0.46875	1.18858799
10	0.68151815	0.90252956	0.56770833	1.08376584
11	0.66940789	0.90300523	0.59375	0.99102196
12	0.69306931	0.85301485	0.63541667	0.91328507
13	0.69078947	0.83639649	0.66145833	0.89352692
14	0.7310231	0.80949896	0.65625	0.87642574
15	0.71217105	0.83083739	0.65104167	0.88518055
16	0.7310231	0.79042478	0.65104167	0.90074011
17	0.75822368	0.75414016	0.63020833	0.86114474
18	0.73190789	0.73097902	0.66145833	0.85052638
19	0.73927393	0.78517573	0.66145833	0.83963467
20	0.7640264	0.70506842	0.65625	0.87778192

21	0.76151316	0.68989575	0.69791667	0.80441864
22	0.76072607	0.71287948	0.625	0.92252454
23	0.79276316	0.66613969	0.65625	0.92212395
24	0.7689769	0.69032831	0.70833333	0.79595708
25	0.76644737	0.67383991	0.66666667	0.89957176
26	0.75742574	0.69896581	0.71354167	0.79330927
27	0.76809211	0.64033233	0.64583333	0.87467607
28	0.78453947	0.66381058	0.69791667	0.80162918
29	0.77483444	0.62891007	0.609375	1.07188962



[그림 4-2] InceptionV3 학습 그래프

4.2.3 ResNet50

[표 4-6] ResNet50 기본 설정 값

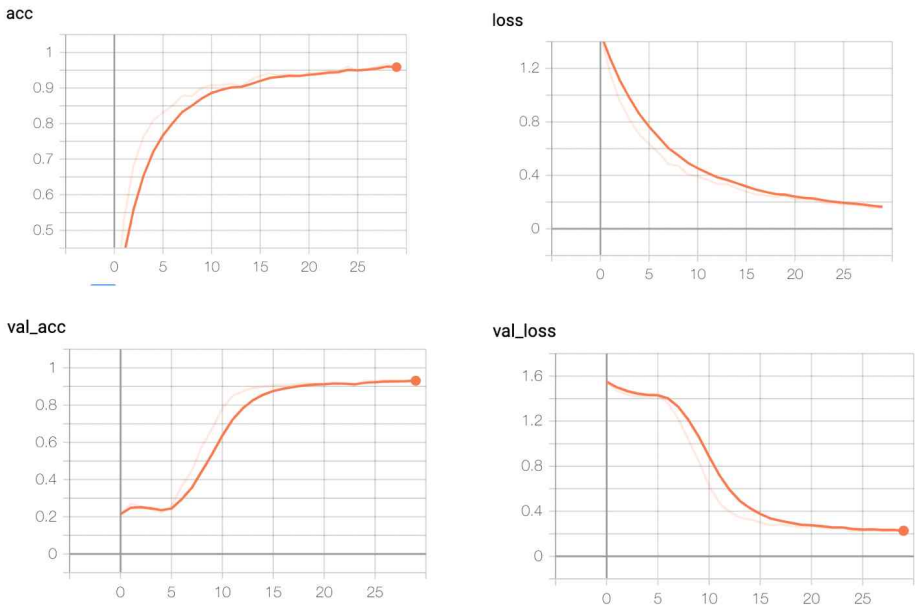
항목	1차 학습	2차 학습
loss	categorical_crossentropy	categorical_crossentropy
optimizer	SDG	SDG
learning rate	0.00004	0.00004
momentum	0.9	0.9
training time	2시간	6시간 40분
accuracy	88%	95.69%
validation accuracy	17%	93.45%
test accuracy	-	92%

ResNet50모델의 총 학습시간은 2시간이 소요되었으며, 결과는 다음 표와 같이 최종 epoch에서의 accuracy는 88% 로 나왔으며 validation accuracy는 17%로 나왔다. InceptionV3 모델과 같이 validation accuracy가 현저히 낮게 학습되어 사전 학습 모델의 가중치를 유지하기 위해 재학습 하지 않게 설정되어 있던 부분을 재학습할 수 있게 설정 후 2차 학습을 진행하였고 그 결과는 95% 이상의 정확도를 보여주었다.

[표 4-7] ResNet50 모델 학습 결과

epoch	acc	loss	val_acc	val_loss
0	0.26917058	1.46393963	0.21410579	1.54991737
1	0.53051643	1.16261784	0.26952141	1.46891116
2	0.68388106	0.95389599	0.25440806	1.43564718
3	0.76212833	0.82022952	0.23929471	1.41680108
4	0.80877743	0.70517939	0.22166247	1.41799482

5	0.83125	0.63355645	0.25692695	1.4257517
6	0.85109718	0.56814339	0.36523929	1.35933031
7	0.87890625	0.48516285	0.44584383	1.21839041
8	0.87774295	0.47182734	0.57934509	1.03466975
9	0.90062598	0.40874963	0.67254408	0.84196809
10	0.9084507	0.39271485	0.78337532	0.6269835
11	0.90766823	0.36790526	0.8488665	0.47453792
12	0.91236307	0.33599842	0.87405542	0.39605162
13	0.90453834	0.337036	0.89168766	0.34474557
14	0.92253521	0.30490348	0.89924433	0.32838763
15	0.9342723	0.27866541	0.90680101	0.30071467
16	0.94053208	0.25812684	0.90680101	0.27568333
17	0.93583725	0.25034572	0.91183879	0.28295128
18	0.93896714	0.23684808	0.91687657	0.27083053
19	0.93338558	0.24801351	0.91687657	0.25618415
20	0.9421875	0.21809862	0.91435768	0.26831661
21	0.94278997	0.21926946	0.92191436	0.25513973
22	0.94835681	0.21759565	0.91435768	0.243709
23	0.94679186	0.19184408	0.90680101	0.25519332
24	0.9600939	0.18626611	0.93198992	0.22359707
25	0.94765625	0.18329379	0.92695214	0.22987469
26	0.95461659	0.18108951	0.93450882	0.23965176
27	0.95924765	0.1700587	0.92947103	0.22602525
28	0.96791862	0.15699498	0.92947103	0.23564639
29	0.95696401	0.15535142	0.93450882	0.21643222



[그림 4-3] ResNet50 학습그래프

각 모델별 test accuracy 측정은 모두 동일한 환경에서 진행하였다.

테스트 데이터셋에서 랜덤하게 8개의 이미지를 추출 한 후 해당 이미지의 예측값과 실제 정답값을 비교하여 평균을 구하였으며, 랜덤 이미지 추출 시 동일한 결과를 보여주기 위해 seed 값을 똑같이 설정하였다. 총 3번의 테스트를 진행하여 평균값을 계산하여 본 결과 ResNet50 이 제일 높은 정확도를 보여주었다.

4.3 테스트 데이터 셋을 이용한 어종별 실험

4.2절을 통해 비교 실험한 결과 제일 높은 정확도를 보여준 ResNet50에 대해 어종 별 테스트 데이터 셋을 이용한 정확도 실험을 하였으며, 실험 결과는 [표 4-8] 과 같다. 실험 방법은 각 어종별로 준비된 테스트 데이터 100장 중 랜덤하게 30개를 추출하여 정답 값과 학습 모델의 예측 값

을 비교하여 계산하였다.

[표 4-8] 어종 별 정확도

참돔	우럭	배스	쏘가리
97%	100%	90%	87%

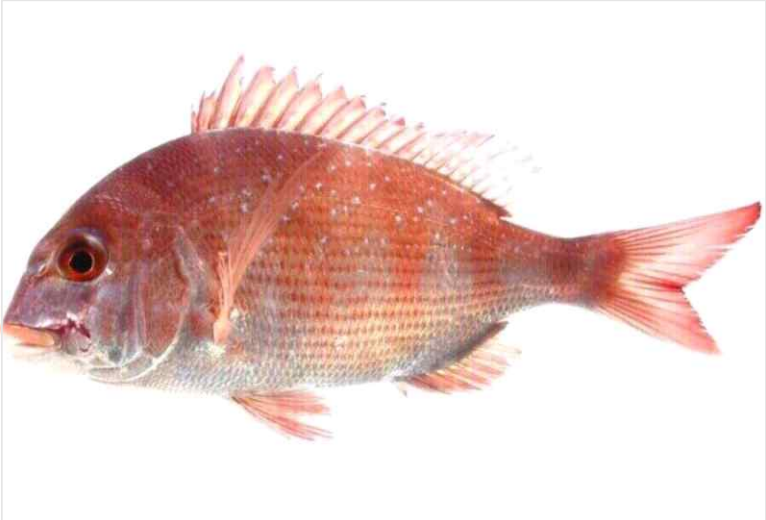
실험한 4종의 어종 중 배스와 쏘가리의 정확도가 각각 90%와 87%로 97%와 100%인 참돔과 우럭보다 다소 떨어지는 결과를 보여주었다.

4.4 스마트폰을 이용한 프로토타입 실험

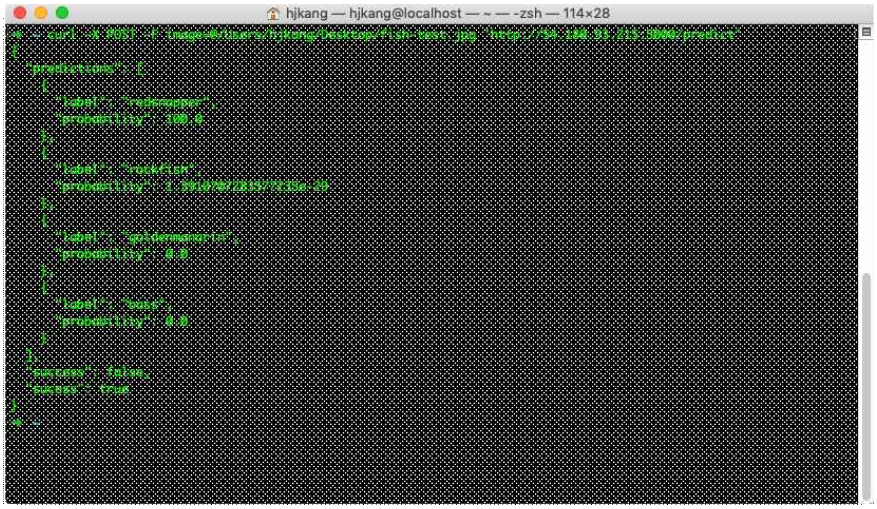
4.2절을 통해 비교 실험한 결과를 토대로 최적의 모델과 조건을 최종 서버에 모델을 배포 하여 로컬 테스트 환경이 아닌 스마트폰과 실제 서버 통신을 통한 어종 판별 테스트를 통해 정답률 등을 실험한다.

네트워크 모델은 비교 실험에서 가장 정확도가 높게 나왔던 ResNet50 모델과 가중치 데이터를 저장하여 서버에서 불러올 수 있게 구성하였고 조건은 4.2절에서와 동일한 조건으로 서버에 배포하였다.

서버로 호출하는 API와 응답 값은 [그림 4-2] 과 같다. 호출 테스트에 사용한 fish-test.jpg 이미지[그림 4-1]의 실제 정답은 참돔으로 서버에서 판별한 결과 값도 redsnapper(참돔 label) 100%로 일치함을 알 수 있다.

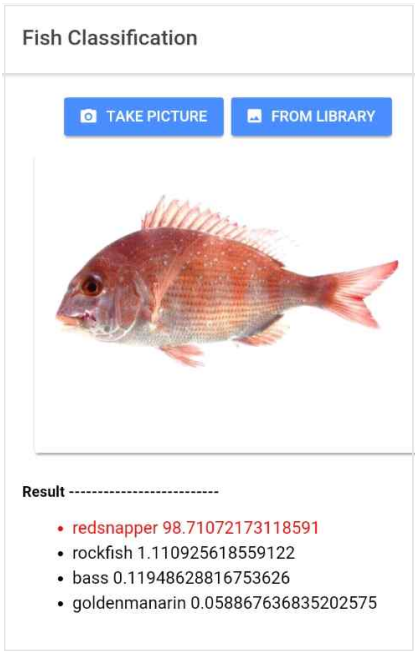


[그림 4-4] fish-test.jpg 이미지 (참돔)



[그림 4-5] 서버 호출 및 응답 결과 화면

위와 동일한 테스트를 프로토타입으로 개발된 어플리케이션에서 진행하면 [그림 4-3]과 같이 동일하게 참돔으로 결과 값을 나타낸다.



[그림 4-6] App 테스트 화면

제 5 장 결 론

본 논문에서는 전이학습을 통해 적은 양의 데이터로 CNN을 통한 학습을 진행하였으며, 스마트폰을 이용한 시스템을 구축하여 어종판별을 손쉽게 할 수 있는 방법을 제안하였다. 전이학습에는 사전 학습된 여러 모델을 비교 실험하였으며, 최종적으로 95.69%의 정확도로 최적의 성능을 보여준 ResNet50 모델을 사용하였다.

사전 학습된 각 모델별 레이어 구조나 epoch 수, batch 사이즈 등 다양한 초기 값에 따라 전이 학습 시 학습결과가 상이하게 나타났으며 최적의 결과 값을 얻기 위해 각 모델별로 학습을 진행하여 3가지 실험한 모델 모두 80%이상의 정확도를 보여주는 모델로 구축하였다.

스마트폰에서 사진을 촬영하거나 선택하여 서버에 전송하면 서버에서 전송받은 이미지를 모델에 맞는 Input 사이즈로 조절한다. 본 논문에서는 사용된 ResNet50의 Input 값에 맞게 224x224 로 리사이징 하도록 전처리 작업을 진행하였다. 실제 학습된 결과는 정확도 95.69%를 보여주었으나, 스마트폰을 이용하여 학습에 사용하지 않은 새로운 어종 사진을 테스트해본 결과 학습에 사용한 4종의 어종 중 참돔은 97%, 우럭은 100%로 비교적 높은 정확도를 보였으며, 쏘가리는 87%이며, 배스는 90%로 쏘가리와 배스의 경우 다소 정확도가 떨어짐을 확인하였다. 이는 쏘가리와 배스의 특징점이 잘 구분되지 않아서 발생할 수 있다고 판단되었다. 추후에 더 많은 데이터를 가지고 epoch을 늘려보는 등 성능 향상을 위해 학습이 더 진행한다면 더 정확한 판별은 수행할 수 있을 것이며, 이후 연구에는 어종의 종류도 더 추가하여 진행해 보고자 한다.

참고문헌

[1] 오타 미쯔히사, 수도 코다이, 쿠로사와 타쿠마, 오다 다이스케 (2019). **실전! 딥러닝**: 텐서플로와 케라스를 이용한 딥러닝 최신 기술 활용 가이드 (손민규, 역). 파주: 위키북스.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. (2012)

[3] HE, Kaiming, et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 770-778.

[4] 박진현, 황광복, 박희문 and 최영규(2019), "어종 분류를 위한 CNN의 적용," *한국정보통신학회논문지,* 23(1), 39-46.

[5] 강종호, 이정삼, and 이헌동. "우리나라 낚시산업의 부가가치 창출방안에 관한 연구." *연구보고서* (2011): 1-255.

[6] 임현교, 김주봉, 권도형, 한연희. “MNIST의 CNN모델 기반 TensorFlow Optimizer 성능 비교 분석”. Advaced Technology Research Center Korea University of Technology and Education

[7] “딥 러닝-위키백과”. 위키백과, https://ko.wikipedia.org/wiki/%EB%94%A5_%EB%9F%AC%EB%8B%9D

[8] “수산생명자원정보센터”. 국립수산과학원, https://www.nifs.go.kr/frcenter/species/?_p=species_view&mf_tax_id=MF0008712

[9] “수산생명자원정보센터”. 국립수산과학원, https://www.nifs.go.kr/frcenter/species/?_p=species_view&mf_tax_id=MF0000569

[10] “전자생물도감 | 해양수산자원연구소”. 경기도 해양수산자원연구소, https://fish.gg.go.kr/data/30?c_order=o9&c_pid=318
[11] “큰입우럭 - 나무위키”. 나무위키, https://namu.wiki/w/%ED%81%B0%EC%9E%85%EC%9A%B0%EB%9F%AD.
[12] “Activation Functions-Jorge Leonel”. Medium, https://medium.com/@jorgesleonel/activation-functions-99fe10770c77
[13] “[Part V. Best CNN Architecture]6.VGGNet[1]-라온피플머신러닝 아카데미-”, https://m.blog.naver.com/PostView.nhn?blogId=laonple&logNo=220738560542&categoryNo=22&proxyReferer=&proxyReferer=https%3A%2F%2Fwww.google.com%2F