CNN 전이 학습을 이용한 고양이 종 판별 프로젝트

Cat Species Classification with CNN Transfer Learning

요 약

이미지 분석에 있어서 CNN이 뛰어난 성능을 입증했으며 그 예로, RsNet, GoogleNet가 있다. 이중 GooLeNet 의 Inception은 네트워크가 깊어지면 깊어 질수록 학습하는 파라미터의 수가 늘어나는 문제를 획기적으로 해결하고 성능을 높인 모델이라고 할 수 있다. Deep Learning guinea pig image classification using Nvidia DIGITS and GoogleNet 이 논문은 본 보고서와 마찬가지로 Inception 모델을 이용하여 guinea pig를 주변환경과 구분하여 classification을 구현한 논문이다. 두 논문의 이론적인 부분과 Tensorflow에서 Inception v3를 이용하여 Image classification 전이 학습으로 이용할 수 있는 Open Source를 제공하는데 이를 통해 고양이 8종을 분류 할 수 있는지를 확인하는 프로젝트를 진행하고자 한다.

1. 서 론

최근에 많은 사람들이 반려동물을 키우고 있다. 이러한 반려 동물의 대표주자로는 고양이, 강아지, 물고기 등이 있다. 이 중 고양이의 경우는 인기가 많아서 많은 사람들이 반려동물로 선택을 하고 있지만 종의 특징을 정확하게 알고 있는 이는 많지 않다. 이러한 고양이의 종을 이미지를 통해서 특정을 지을 수 있다면 좀 더 많은 사람들이 고양이에 대하여 관심을 가질 수 있다고 생각이 들어서 이번 프로젝트를 진행하게 되었다.

프로젝트를 진행하기 위해서 Google Inception Model[1]을 전이학습을 통해서 이용하기로 하였다. 마침 Tensorflow에서 Imagine Retrained라는 Open Source로 이를 쉽게 사용 할 수 있는 코드가 존재하였고 활용하는 방법도 Google에 많이 나와 있어 어렵지는 않았다. 또한 비슷한 연구로는 Deep Learning guinea pig image classification using Nvidia DIGITS and Google Net[2]이라는 논문이 있어서 이를 참조하면서 프로젝트를 진행하고자 한다.

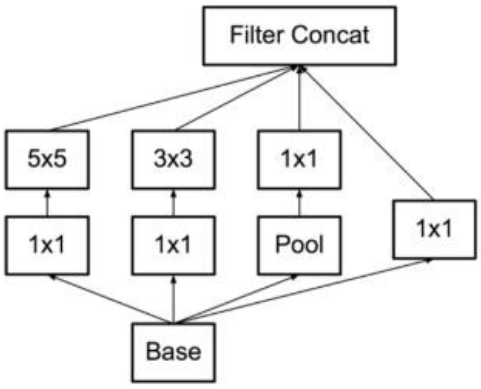
프로젝트의 목표는 수 많은 고양이 중 일단 8가지의 종류의 고양이를 구분하는 모델을 만들어 보고자 한다. 8가지 종으로는 Persian, Scottish Fold, American Short Hair, British Short Hair, Russian Blue, Munchkin, Ragdoll, Siamese을 사용하고 종 분류 정확도는 85%를 목표로 한다.

2. 기존 연구

2.1 Rethinking the Inception Architecture for Computer Vision

이 논문은, 2014년에 처음 나온 Inception v1의 내용을 좀 더 많은 사람들이 쉽게 접근하고 사용할 수 있도록 개량을 한 Inception v2와 v3에 관한 논문이다.

먼저 Inception Module에 대하여 간략하게 소개를 하자면 Inception Module은 feature를 효율적으로 추축하기 위해서 1x1, 3x3, 5x5의 convolution 연산을 각각 수행한다. 이때 1x1 convolution 을 이용하여 각 3x3, 5x5의 연산량을 줄일 수 있고 Max-Pooling의 경우 1x1이 뒤에 있는데 이유는 Max-Pooling은 C의 크기 조절이 불가능하기 때문에 출력 C를 맞추기 위해서 뒤에 존재하게 된다.



original inception module

이러한 Inception 모델의 경우 기존의 모델보다 효과적으로 파라미터의 수와 차원을 줄여 연산량을 줄일 수 있었지만 이를 변형시켜서 응용하는 것은 쉽지 않았다. 또한 기존의 VCG가 사용한 3x3 conv 필터만 이용하는 것이 효과가 여전히 좋았기 때문에 많이 활용되지

못했다.

이를 해결 하기 위해서 구글은 기존의 Inception Module을 개량한 v2와 v3를 내놓게 된다.

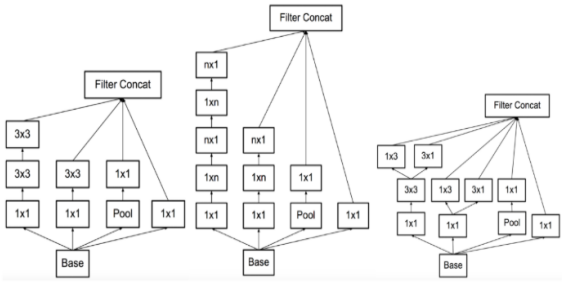


Fig. 5. 3 x Inception-A Fig. 6. 5 x Inception-B Fig. 7. 2 x Inception-C

Factorizing된 Inception Module은 기존의 5x5 convolution Filter 대신 3x3 Filter를 2개를 이용하는 방식을 사용하였다. 뿐만 아니라 비대칭 Conv을 이용하여 연산량을 더 줄일 수 있었고 Feature Map의 Grid를 줄이기 위해서 Pooling과 Conv의 순서를 결정 해야 했는데 차이가 존재하지 않아 이를 병렬로 섞어 버리는 모델을 만들었고 그것이 가장 오른쪽에 있는 모델이다. 이렇게 만들어진 것이 v2 모델이었고 이를 더 개량한 것이 v3 모델이다. V3의 경우는 Optimizer를 RMSProp로 변경, 마지막에 F-C layer에 Batch Normalization을 적용, Label을 one-hot encoding이 아닌 값을 1-(n-1)*e으로 적용하여 값이 0인 레이블에도 값을 적용하는 것이다. 이를 통해서 error 발생률을 3.58%까지 줄일 수 있었다.

2.2 Deep Learning guinea pig image classification using Nvidia DIGITS and GoogleNet

이 논문에서는 pre-trained 된 모델들을 이용하여서 guinea pig를 구분하는 모델을 만들고자 했다.

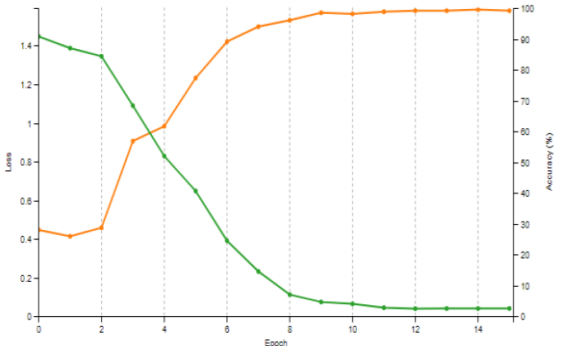
Fig. 2. Example images for crested, abyssinian and skinny guinea pigs from the provided data set.

총 이용된 Data set 은 1098개의 image를 이용하여서 진행을 하였는데 25%인 274장은 검증에, 10% 110장은 test용으로 활용하였다. 사용된 이미지의 관점은 총 4가지였는데 입 모양이 나오는 얼굴, 얼굴의 정면과 후면의 모습, 전체적인 동물의 뒷모습 그리고 위에서 보는 경우 거리에 차이를 둔 시점으로 두었다.

총 3가지의 실험을 진행 했으며 결과는 다음과 같다.

1. **First test:**
   - Epoch count: 15,
   - Learning rate: 0.001 with fixed policy,
   - Optimizer: Stochastic Gradient Descent.
2. **Second test:**
   - Epoch count: 15,
   - Learning rate: 0.001 with step down policy,
   - Optimizer: Adam.
3. **Third test:**
   - Epoch count: 15,
   - Learning rate: 0.01 with step down policy,
   - Optimizer: Stochastic Gradient Descent.

아래의 그래프는 2번째 테스트의 결과이다.



최종적으로 99.31% 정확도와 0.04의 loss 값을 얻을 수 있었다. 이 두 번째 논문에서 실험하는 방식이 이번 프로젝트의 진행방식과 매우 비슷한 전형적인 전이 학습을 이용한 방식이다. 나 또한 기존에 학습 되어 있는 Inception Module을 가져와서 논문 2와 같이 실험을 하고자 한다.

3. 시스템 모델

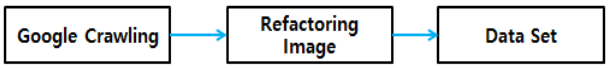
3.1 기존 연구와 차이점 및 해결방안

기존의 논문과 다른 점은 바로 과업에 있어서 도메인의 차이인 것 같다. 현재 기존의 이미지 학습 모델을 이용하여 다양한 전이 학습이 이루어지고 있는데 이중 나는 고양이 종 이라는 특정 도메인에 초점을 맞추었다. 때문에 단순히 기존에 있는 모델에다가 네트워크의 하단

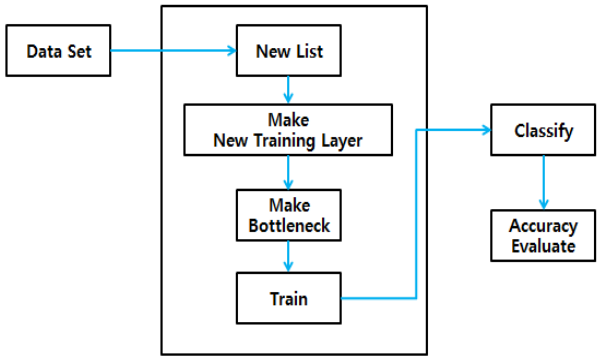
부분의 Fully Connected Layer만 수정을 하면 되는 Fine Tuning 작업을 이용할 것이기 때문에 도메인의 차이가 가장 큰 차이점이다.

이러한 도메인 차이를 해결하기 위해서 자료를 모우던 도중 Tensorflow에서 Open Source로 제공하는 Image Retrained 코드를 찾을 수 있었고 이를 기존의 Inception Model과 결합하여 프로젝트를 진행 하고자 한다.

4. 프로젝트 진행 방식



이 프로젝트는 크게 2가지 작업을 통해서 진행을 하였다. 첫 번째 작업은 먼저 모델을 위한 데이터 샘플을 모우기 위해 Python과 Google Crawling function을 이용하여서 8종 고양이에 대한 Image를 수집한다. 수집한 Image의 경우 1차적으로 종에 맞지 않는 이미지나 잘못된 이미지를 수정하거나 제거하여서 Refactoring 작업을 진행한다. 만들어진 이미지들을 Label에 맞게 폴더에 정리한다.



2차 작업으로는 만들어진 Label Folder Data들을 가지고 학습을 진행한다. 이때 기존에 학습된 Inception Module의 파라미터 값을 이용하는 것이 이미지에서 추출한 특징을 Bottleneck화 시켜 작업을 하는 것이다. 그 다음 새로운 도메인 판별을 위한 새로운 F-C layer인 Training Layer를 기존의 Inception v3와 연결하여서 학습을 시작하고 분류를 시작한다. 이후 만들어진 모델을 사전에 분류해둔 test 이미지를 통해서 정확도를 측정한다.

5. 프로젝트 실험

프로젝트 진행은 위에서 만들어진 2가지 단계를 이용하여 다음과 같은 2가지 데이터 셋을 Feed 데이터로 넣어서 그에 대한 정확도를 측정하는 것으로 진행을 하였다.

Train1 (Total Sample 2349)

Train1 Data Sample

	American Short	British Short	Munchkin	Persian	Ragdoll	Russian Blue	Scottish Fold	SIAMESE	Total
Count	309	291	217	279	230	369	352	302	2349

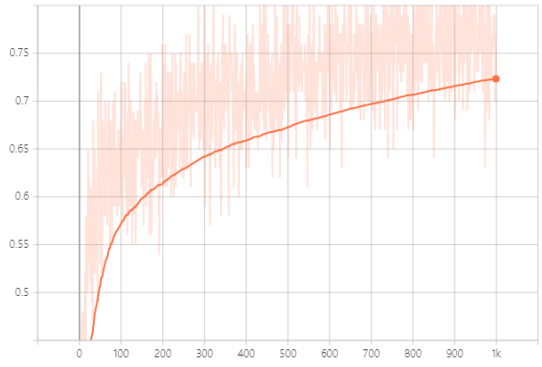
Train1 Setting

	Optimizer	Learning rate	Loss function	Train Step	Batch Size
value	Gradient Descent	0.01	Softmax Cross Entropy	1000	Train: 100 Validation :100

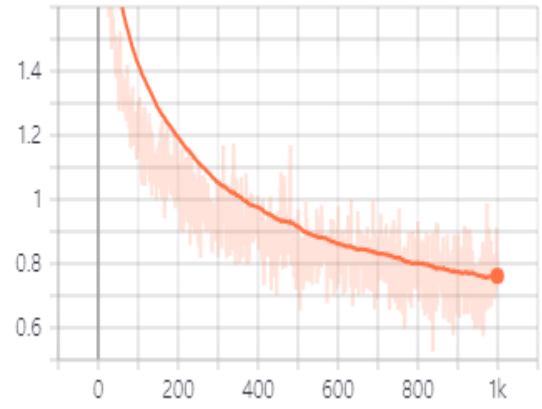
Train 1 Setting 상태이다. Optimizer로는 Gradient Descent Optimizer를 사용, Loss Function으로는 softmax_max cross_entropy with logit 함수를 이용하여서 실행하였다. Learning rate은 0.01, 학습 빈도는 1천번으로 실행 하였으며 Batch Size의 경우는 100으로 설정했다.

학습 결과는 Tensorflow에서 제공하는 Tensor board를 이용하여서 그래프로 나타내 보았다. 그래프를 편리하기 보기 위해서 Smoothing을 0.99로 설정하여서 나타내었다.

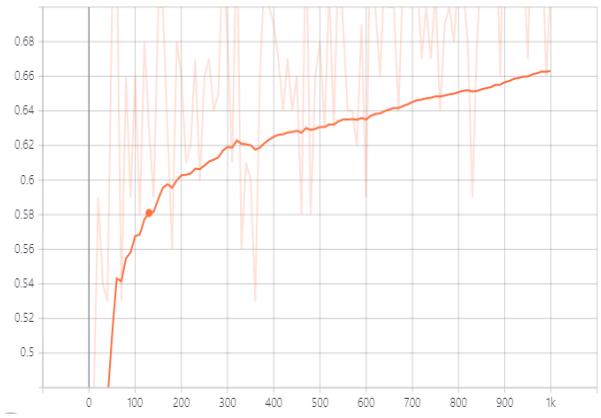
Train1 Train Accuracy (Smoothing 0.99)



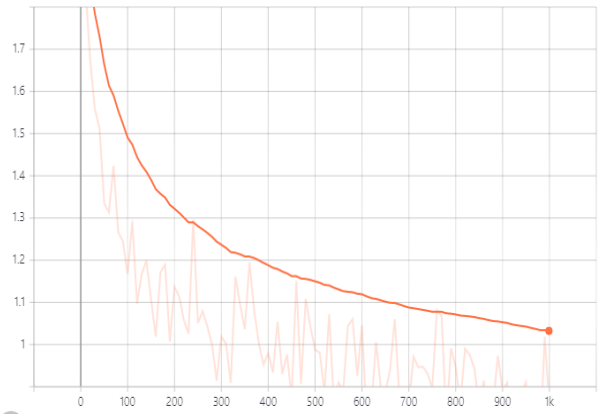
Train1 Train Cross Entropy (Smoothing 0.99)



Train1 Validation Accuracy (Smoothing 0.99)



Train1 Validation Cross Entropy (Smoothing 0.99)



Train2 (Total Sample 3005)

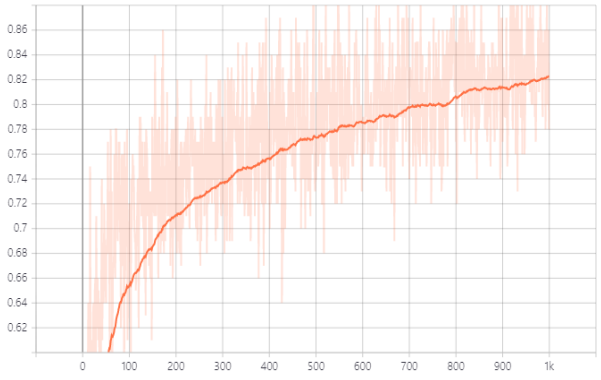
Train2 Data Sample

	American Short	British Short	Munchkin	Persian	Ragdoll	Russian Blue	Scottish Fold	SIAMESE	Total
Count	304	415	306	407	401	420	402	350	3005

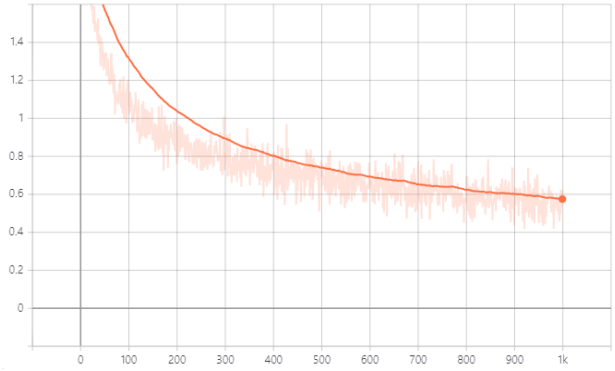
Train2 Setting

	Optimizer	Learning rate	Loss function	Train Step	Batch Size
value	Gradient Descent	0.01	Softmax Cross Entropy	1000	Train: 100 Validation :100

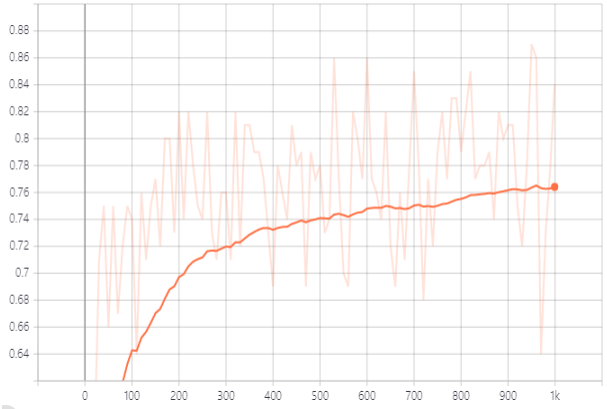
Train2 Train Accuracy (Smoothing 0.99)



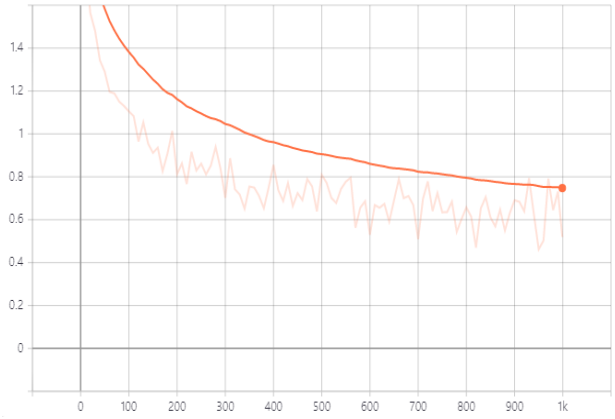
Train2 Train Cross Entropy (Smoothing 0.99)



Train2 Validation Accuracy (Smoothing 0.99)



Train2 Validation Cross Entropy (Smoothing 0.99)



Result (Final step (999) 기준)

Train1	Accuracy	Cross Entropy
Train1 – Train	72%	0.8009
Train1 – Val	71%	0.8687

Result (Final step (999) 기준)

Train2	Accuracy	Cross Entropy
Train2 – Train	78%	0.6195
Train2 – Val	84%	0.5192

최종 결과가 나온 값을 보면 알 수 있지만 Train2의 경우 Train1 보다 전체적으로 정확도가 올라 갔고

Cross Entropy의 값 즉 loss Function의 값이 더 작아 졌음을 알 수 있다. 이를 하나의 표로 나타내면 다음과 같다.

Comparison Result

	Accuracy	Cross Entropy
Train	1.08%	0.77%
Validation	1.18%	0.59%

이는 더 정확하고 많은 데이터가 들어가서 좀 더 좋은 모델이 만들어 졌다는 것을 의미한다.

6. 모델 검증 및 결과 분석

그렇다면 실제로 만든 모델이 제대로 작동하는지를 파악하기 위해서 나는 훈련과 검증에 사용되지 않은 전혀 새로운 고양이 사진을 이용하여서 이를 테스트 해보고자 한다.

실제 이미지를 집어 넣었을 때 해당 고양이에 대한 Scoring하게 되고 이 때 그 점수가 얼마인지를 소수점 3번째 자리에서 반올림 하여서 기록하였다.

실험에 사용되는 고양이 사진은 총 종마다 5장이고 Random하게 선정을 하였다.

Train1 Result

	1	2	3	4	5
American	0.83	0.86	0.89	0.69	0.81
British	0.34	0.76	0.52	0.66	0.65
Munchkin	0.05	0.20	0.28	0.28	0.30
Persian	0.89	0.91	0.78	0.72	0.53
Ragdoll	0.90	0.97	0.84	0.40	0.72
Russian	0.97	0.96	0.86	0.90	0.87
Scottish	0.50	0.59	0.25	0.37	0.36
Siamese	0.98	0.88	0.85	0.99	0.99

Train2 Result

	1	2	3	4	5
American	0.89	0.84	0.67	0.89	0.95
British	0.53	0.67	0.92	0.84	0.80
Munchkin	0.20	0.23	0.24	0.60	0.67
Persian	0.93	0.95	0.68	0.57	0.85
Ragdoll	0.95	0.90	0.40	0.65	0.95
Russian	0.97	0.94	0.70	0.79	0.58
Scottish	0.50	0.76	0.38	0.32	0.32
Siamese	0.98	0.81	0.85	0.99	0.99

결과를 비교한 결과 대부분의 점수에서 Train2에서 미세하게 또는 좋은 결과를 얻을 수 있다. 물론 오히려 역으로 판별을 못하는 경우도 존재하였기 때문에 완벽하게 구분을 한다고 할 수는 없을 것 같다.

노란색 부분으로 표시된 Munchkin과 Scottish Fold의 경우는 판별 정확도가 다른 고양이 종에 비해서 매우 떨어짐을 보이고 있다. 반면, 나머지 6종에서 대부분 70%를 넘는 정확도를 보였고 특히, Siamese, Persian, Ragdoll, Russian Blue의 경우는 95%가 넘는 정확도를 측정한 이미지도 있었기 때문에 절반 정도는 성공을 했다고 생각한다. 물론 판정이 잘된 고양이 내에서도 편차가 큰 사진이 있었기 때문에 완벽하다고는 할 수 없다.

6.1 차이가 발생한 원인 분석

Munchkin의 경우에는 나 또한 데이터 분류 및 가공을 할 때 이 사진이 정말 Munchkin이 맞는지 혼동되는 사진이 많았기 때문에 실제로도 모델 학습에서도 그러한 점이 반영이 된 것 같다. Scottish Fold의 경우 이름의 어원에도 알 수 있듯이 고양이 귀 접히는 것이 특징인 고양이인데 귀가 접히는 것은 다른 고양이 이미지에서도 종종 보였기 때문에 그래서 오히려 판별을 하지 못하였다고 생각이 된다.

좋은 값을 도출해낸 고양이의 경우는 명확한 특징이 존재했다. 평균적으로 가장 높은 값을 자랑한 Siamese 같은 경우는 얼굴에 및 귀가 검다는 점과 푸른색 눈동자 그리고 다리 와 꼬리도 검다는 명확한 특징이 존재한다. Russian Blue도 전체적으로 회색 및 남 청색의 털 색과 청안 또는 녹안이라는 명확한 특징이 있었다. 때문에 이런 명확한 특징을 가진 고양이의 경우는 CNN이 Feature를 제대로 잡아서 명확하게 학습을 하였지만 Munchkin이나 Scottish Fold는 그러한 Feature를 잡아내지 못한 것으로 추측을 하고 있다.

7. 결론 및 향후 연구

이 고양이 종 프로젝트에서 사용된 모델의 경우는 데이터 셋의 차이만 존재한다. 때문에 전이학습에서 사용된 모델의 Network구조나 마지막 Fully Connected 영역에서 사용되는 Optimizer, Loss Function, Learning Rate 등 학습을 위한 모델과 관련된 변수 자체는 수정을 하지 않았다. 때문에 이러한 함수 및 변수를 수정하여서 더 많은 모델에서 측정을 해보고 싶다. 특히, 이번 모델을 만들기 위해서 Train Step을 1000으로 두고 하였는데 만약 5000, 10000을 하였다면 더 좋은 결과 값이 나왔을 것이라고 예상 한다. 또한 데이터 셋을 만들기 위해서 Crawling하고 그 다음 주관적으로 데이터를 판별하고 편집을 비전문가인

내가 했기 때문에 데이터의 객관성에도 문제가 있다고 생각을 하고 있다. 전문가가 데이터를 셋을 만드는데 협력을 한다면 더 좋은 데이터 셋을 만들 수 있고 이는 모델에도 영향을 줄 것이라고 생각한다.

마지막으로 이 모델이 제대로 학습이 되어서 일반화가 명확하게 된다면 학습된 모델의 값을 Freeze 시켜서 다른 동물의 종에도 사용을 해보고 싶다. 특히, 고양이와 항상 비교되는 개에 대입을 한다면 어떠한 결과가 나올지 궁금하다. 또한 순수한 고양이 종만 판별하는 것이 아니라 혼혈인 고양이의 종도 분류하는 모델을 향후 과제로 삼아서 연구를 진행해보고 싶다.

참 고 문 헌

[1] Christian Szegedy, Vicent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Google Inc, Zbigniew Wonjna University College London.
Rethinking the Inception Architecture for Computer Vision. (Submitted on 2 Dec 2015 (v1), last revised 11 Dec 2015 (this version, v3)

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Google Inc, University of North Carolina Chapel Hill, University of Michigan.
Going deeper with convolutions
(Submitted on 17 Sep 2014)

[3] Lukasz Zmudzinski, University of Warmia and Mazury in Olsztyn, Poland.
Deep Learning guinea pig image classification using Nvidia DIGITS and GoogLeNet
September 2018, Conference: 27th international Workshop on Concurrency, Specification and Programming (CS&P'18), At Berlin, Germany

[4] 텐서 플로우(TesorFlow)를 이용해서 Inception V3 모델 Retrainging을 통해 나만의 데이터 셋을 이미지 인식(추론) 해보기 http://solarisailab.com/archives/1422

[5] Tensorflow – Image Retraining
https://github.com/tensorflow/hub/tree/master/examples/image_retraining