

2018 년 2 학기 인공지능 최종 과제 보고서

HUMAN PROTEIN ATLAS IMAGE CLASSIFICATION

조 TEAM 설명

조 이름 : KMU_AI_ZEKUS

학번_이름 : 20132651_이성재

학번_이름 : 20130940_장용훈

수행한 업무 분장

이성재 (50%) : AWS 분석환경 세팅 / 모델 수정 및 튜닝 / 검증 및 최종제출

장용훈 (50%) : 데이터 전처리 및 확대기법 적용 / 제안서 및 보고서 작성

주제 설명 및 EDA

먼저 주어진 문제에 대해 살펴보면 다음과 같습니다. 사람의 단백질은 28 가지 정도의 소기관으로 구성되어 있으며, 구성을 판단하기 위해 우리는 단백질, 핵세포, 미세 소관, 소포체 등의 형태를 확인해야 합니다. 하나의 단백질 패턴 속에 여러 종류의 소기관 구성이 나타날 수 있으며, 이러한 소기관 구성에 대한 분류를 컴퓨터가 예측할 수 있도록 만드는 것이 이번 문제의 핵심입니다.

주어진 단백질 패턴은 몇 가지 특징을 가지고 있습니다. 단백질 소기관의 분포를 순서대로 정렬하였을 때, 상위 소기관들의 빈도수가 높고 (Nucleoplasm 의 경우는 30%를 차지) 하위 소기관의 경우 빈도수가 낮은 (Rods & Rings 는 0.1% 차지) 데이터 불균형 현상이 나타납니다. 따라서 학습을 모델링할 때 이러한 불균형을 고려해야만 합니다. 다른 특징으로는 하위 소기관 단독으로 구성되는 경우가 48%, 2 개로 구성되는 경우가 40%, 3 개 이상으로 구성된 경우가 12% 정도로 분포한다는 점입니다. 하나의 단백질 특징이 1~2 개 정도의 소기관 구성을 가진다는 것입니다.

데이터가 주어지는 형태에 대해 살펴보면, 일반적으로 볼 수 있는 RGB 의 3 채널 이미지가 아닌 RGBY 의 4 채널 이미지로 제공됩니다. 각각의 채널은 단백질, 핵세포, 미세소관, 소포체의 형태를 나타내며, 4 개 채널을 모두 합쳤을 때 하나의 완전한 단백질 패턴 이미지가 나타납니다. 가장 중요한 이미지는 Green 채널의 단백질 이미지이며, 나머지 R, B, Y 이미지는 단백질 패턴의 구성을 예측하는데 도움을 주기 위한 보조 데이터의 역할을 합니다. 최종적인 출력은 해당 단백질 패턴의 id 와 단백질 패턴이 보유한 소기관의 번호를 예측한 y 값으로 나타내는 것이며, 여러개의 y 값을 출력해야 한다는 점에 의해 각 클래스에 Threshold 등을 지정해야 하는 Multilabel Classification 임을 알 수 있습니다.

적용한 기계 학습 상세 및 구현 방법

i. 세부 설명

기계 학습을 적용하기 이전에, 프로젝트 수행 계획을 생각하는 단계에서 Scratch 부터 학습을 진행하기 보다는 Transfer Learning 을 이용하기로 결정했습니다. 이는 3 만장이 넘는 512 x 512 크기의 고화질 데이터를 빠른 시간 안에 학습시키고, 튜닝하기 위함입니다. Transfer Learning 을 하기 위해서 적절한 모델을 찾을 필요가 있었는데, 특정 커널에서 Fast Ai 패키지의 ResNET-34 모델을 이용한 것을 보고 참고하여 작성하게 되었습니다. 하지만 여기서도 입력 채널이 3 채널인 ResNET 을 4 채널인 Protein 이미지 데이터에 사용하기 위해서는 약간의 수정이 필요했습니다.

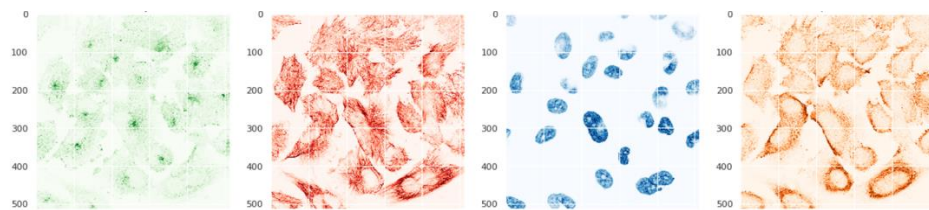


그림 1. RGBY 4 채널 이미지를 채널별로 출력한 모습

채널이 맞지 않는 문제를 해결하기 위해서 기존의 3 채널 ResNET 모델에 1 채널을 추가하는 방법을 사용하였습니다. 이 때, Weight 가 0 으로 주어지는 채널이 아닌, 기존 채널의 Weight 에서 복사해오는 방법을 사용하여 성능 향상을 할 수 있었습니다.

```
w = layers[0].weight
layers[0] = nn.Conv2d(4,64, kernel_size=(7,7), stride=(2,2), padding=(3, 3), bias=False)
layers[0].weight = torch.nn.Parameter(torch.cat((w,w[:,1,:,:,1]),dim=1))
```

그림 2. Weight 를 기존의 weight 에서 복사한 다음, 덧붙여 채널을 늘리는 코드

그 외에도 RandomRotate / RandomDihedral / RandomLighting 과 같은 확대기법을 적용하고, Normalize 하여 데이터를 전처리 하는 과정이 있었습니다. Normalize 를 위해서 필요한 통계치를 전체 데이터에서 가져오고, 이를 활용하여 전처리를 진행하였습니다.

```

x_tot = np.zeros(4)
x2_tot = np.zeros(4)
for x,y in iter(md.trn_dl):
    tmp = md.trn_ds.denorm(x).reshape(16,-1)
    x = md.trn_ds.denorm(x).reshape(-1,4)
    x_tot += x.mean(axis=0)
    x2_tot += (x**2).mean(axis=0)

channel_avr = x_tot/len(md.trn_dl)
channel_std = np.sqrt(x2_tot/len(md.trn_dl) - channel_avr**2)
channel_avr, channel_std

```

그림 3. 전처리를 위해 필요한 수치를 이미지 데이터에서 계산하는 코드

이미지 확대를 비롯하여 규제기법 등이 학습의 성능에 큰 영향을 미치는데, Fast Ai 의 ResNET 에서는 학습기 내부에 다양한 설정이 이미 존재했습니다. 예를 들어 AdaptiveConcatPool2d, DropOut, ReLU, BatchNorm1d 등이 있었으며, 실제 코드에서는 확인할 수 없습니다. 다만 Learner.fit 을 통해 자동적으로 수행하는 AutoML 의 일부라고 볼 수 있겠습니다. 학습기의 전체적인 모습을 확인하면 다음과 같습니다.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

그림 4. 학습에 사용된 ResNET-34 모델의 구조

ResNET 을 이용한 전이학습에서 성능을 높일 수 있었던 가장 큰 요인은 원본 이미지에 가까운 데이터를 사용하는 것 이었습니다. 시간을 단축시키기 위해 256 x 256 크기의 이미지를 사용하는 것은 0.45 ~ 0.46 정도의 성능이 나타났지만, 이를 512 x 512 크기의 이미지로 변화시켰을 경우 최대 0.49 이상의 성능이 나타나는 것을 확인하였습니다. 이렇게 큰 이미지 사이즈를 사용하였을 때 크게 두 가지 문제점이 발생하였습니다. 첫 번째 문제는 SageMaker 의 p2.xlarge 인스턴스는 K80 GPU 를

사용하는데, 여기서 512 사이즈와 64 배치 사이즈를 사용할 경우 Out of memory 에러가 발생합니다. 이를 해결하기 위해서 배치 사이즈를 1/4 인 16 으로 감소시켜 학습을 진행하였습니다. 두 번째 문제는 기존에 1epoch 당 10 분 정도 걸리던 작업이 1epoch 당 50 분 걸리게 증가하였다는 점입니다. 8epoch 를 1 회 학습으로, 총 3 회 학습하는 과정에서 기존의 256 x 256 이미지가 5 시간 정도 걸렸다면, 512x512 이미지는 25 시간 정도 걸리게 되었습니다. 이 문제를 해결하기 위해 기존에 3 회 학습 후 결과를 출력하던 것을, 1 회 학습 후 결과를 출력하도록 변형하였습니다. 그 결과 1 회차 0.47 -> 2 회차 0.48 -> 3 회차 0.49 로 학습량에 따라 성능이 향상되는 모습을 확인할 수 있었습니다.



그림 5. 총 37 회 Submission 을 통해 향상되는 LB Accuracy

앞에서 말했던 데이터의 unbalanced 문제를 해결하기 위해 FocalLoss 를 학습기에 적용하여 이 문제를 해결하였습니다. 깊은 학습에 따른 gradient 폭발 문제를 완화하기 위해서 학습기에 gradient clipping 을 적용하였습니다. 학습률은 일정 epoch 마다 다르게 적용 하였는데, 이는 학습이 학습을 진행 함에 따라 지역점 수렴에 빠지는 것을 빠르게 탈출하기 위해 사용된 것입니다. Layer 마지막으로 score 를 계산할 때는 test time augmentation 을 이용하였습니다.

ii. 구현 환경, 구현 방법

구현 환경을 구축할 때 고려한 몇 가지 사항은 다음과 같습니다. 최소 10 시간 이상의 Stable 한 학습이 가능한가? 최소 34GB (17GB 데이터의 압축 해제) 의 가용 공간이 존재하는가? GPU 를 사용할 수 있는 환경인가? 등 입니다. 이러한 사항을 고려했을 때, 일반적인 Local 환경에서는 구현이 어렵다 생각하고, 최근에 화두가 되는 AWS 의 SageMaker 라는 Cloud 기반의 Machine Learning 환경을 사용하였습니다. AWS 계정은 소프트웨어 학과 사무실을 통해 대여하였으나, 비용을 최소화 하기 위해 가장 저렴한 GPU 인스턴스인 ml.p2.xlarge 를 사용하였습니다. 해당 인스턴스는 1 시간에 약 1.26 달러 정도의 비용이 들며, 24 시간 사용할 경우 30 달러의 비용이 들게 됩니다.

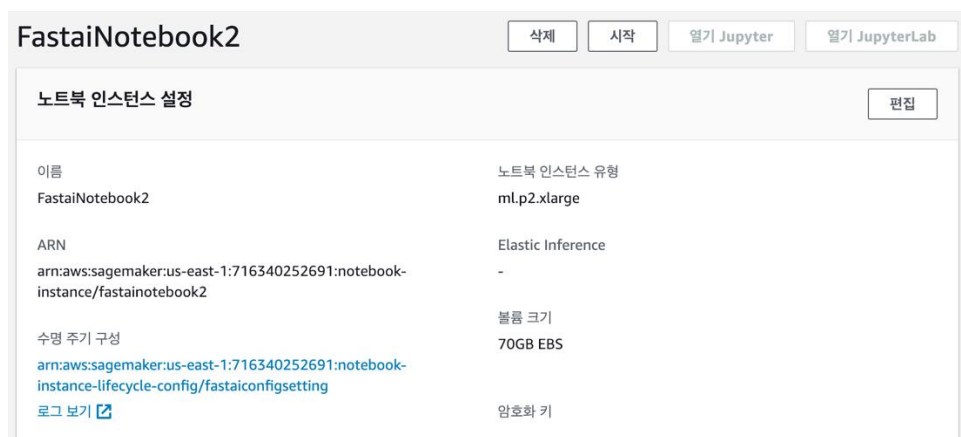













그림 6. Ml.p2.xlarge 유형으로 70GB Storage 를 할당하여 노트북 인스턴스 생성

환경을 구축하는 과정은 다음과 같습니다. 우선 이미지 데이터를 다운로드 받고 결과를 제출하기 위한 Kaggle API 를 설치해야 합니다. API 가 설치되고, zip 파일 형태로 데이터를 받았다면 unzip 을 이용해 압축을 해제해야 합니다. 이 때, unzip 명령어를 사용하기 위해서는 unzip 패키지를 설치해야 하는데, AWS 는 CentOS 를 사용하므로 yum install unzip 을 사용해야 합니다. Kaggle API 설치가 완료되고 데이터를 unzip 하여 접근 가능하게 되면 이 때 부터 Fast Ai 를 이용해 분석과 학습을 진행할 수 있습니다.

결과 : PUBLIC LB 0.496

123	▼ 35	T Flana		0.497	123	1d
124	new	CHAD	   	0.496	8	11h
125	▲ 36	TSMC		0.496	46	2d
126	▼ 5	KMU_AI_ZEKUS	 	0.496	37	5d
Your Best Entry ↑ Your submission scored 0.474, which is not an improvement of your best score. Keep trying!						
127	▼ 9	tito		0.496	16	2d
128	▼ 61	ssshch		0.496	18	1d
129	▼ 63	Alpha Goose		0.495	52	12d

37 submissions for KMU_AI_ZEKUS			Sort by	Most recent				
All	Successful	Selected						
Submission and Description			Public Score	Use for Final Score				
protein_classification10_v.csv 5 days ago by Sungjae Lee add submission details			0.474	<input type="checkbox"/>	protein_classification6_v.csv 9 days ago by Sungjae Lee sixth_submit3		0.442	<input type="checkbox"/>
protein_classification10_05.csv 5 days ago by Sungjae Lee add submission details			0.485	<input type="checkbox"/>	protein_classification6_t.csv 9 days ago by Sungjae Lee sixth_submit2		0.446	<input type="checkbox"/>
protein_classification9_05.csv 5 days ago by Sungjae Lee add submission details			0.491	<input type="checkbox"/>	protein_classification6.csv 9 days ago by Sungjae Lee sixth_submit		0.429	<input type="checkbox"/>
protein_classification10_c.csv 6 days ago by Sungjae Lee 10th_submit3			0.482	<input type="checkbox"/>	protein_classification4_f.csv 10 days ago by Sungjae Lee forth_submit4		0.433	<input type="checkbox"/>
protein_classification9_v.csv 6 days ago by Sungjae Lee 9th_submit4			0.492	<input type="checkbox"/>	protein_classification4_v.csv 10 days ago by Sungjae Lee forth_submit3clear		0.437	<input type="checkbox"/>
protein_classification9_c.csv 6 days ago by Sungjae Lee 9th_submit3			0.496	<input type="checkbox"/>	protein_classification4_v.csv 10 days ago by Sungjae Lee forth_submit3		0.437	<input type="checkbox"/>
protein_classification10_f.csv 6 days ago by Sungjae Lee 10th_submit2			0.481	<input type="checkbox"/>	protein_classification4_t.csv 10 days ago by Sungjae Lee forth_submit2		0.437	<input type="checkbox"/>
protein_classification10_t.csv 6 days ago by Sungjae Lee 10th_submit1			0.474	<input type="checkbox"/>	protein_classification4.csv 10 days ago by Sungjae Lee forth_submit		0.434	<input type="checkbox"/>
protein_classification9_f.csv 7 days ago by Sungjae Lee 9th_submit2			0.485	<input type="checkbox"/>	protein_classification_05.csv 11 days ago by Sungjae Lee 05_test		0.340	<input type="checkbox"/>
protein_classification9_t.csv 7 days ago by Sungjae Lee 9th_submit1			0.494	<input type="checkbox"/>	protein_classification_v.csv 11 days ago by Sungjae Lee v_test		0.363	<input type="checkbox"/>
protein_classification8_f.csv 7 days ago by Sungjae Lee 8th_submit3			0.482	<input type="checkbox"/>	protein_classification_3.csv 11 days ago by Sungjae Lee third_2_submit		0.445	<input type="checkbox"/>
protein_classification8_v.csv 7 days ago by Sungjae Lee 8th_submit2			0.467	<input type="checkbox"/>	protein_classification3.csv 11 days ago by Sungjae Lee third_1_submit		0.451	<input type="checkbox"/>
protein_classification8_t.csv 7 days ago by Sungjae Lee 8th_submit1			0.482	<input type="checkbox"/>				
protein_classification7_f.csv 8 days ago by Sungjae Lee 7th_submit4			0.472	<input type="checkbox"/>				
protein_classification7_05.csv 8 days ago by Sungjae Lee 7th_submit3			0.462	<input type="checkbox"/>				
protein_classification7_v.csv 8 days ago by Sungjae Lee 7th_submit2			0.472	<input type="checkbox"/>	protein_classification_f.csv 11 days ago by Sungjae Lee second_submit		0.385	<input type="checkbox"/>
protein_classification7_t.csv 8 days ago by Sungjae Lee 7th_submit1			0.476	<input type="checkbox"/>	protein_classification_c2.csv 11 days ago by Sungjae Lee second_submit		0.151	<input type="checkbox"/>
protein_classification7_t.csv 8 days ago by Sungjae Lee 7th_submit1			0.476	<input type="checkbox"/>	protein_classification_t2.csv 11 days ago by Sungjae Lee second_submit		0.152	<input type="checkbox"/>
protein_classification6_f.csv 9 days ago by Sungjae Lee sixth_submit5			0.435	<input type="checkbox"/>	protein_classification_c.csv 12 days ago by Sungjae Lee second_submit		0.376	<input type="checkbox"/>
protein_classification6_05.csv 9 days ago by Sungjae Lee sixth_submit3			0.439	<input type="checkbox"/>	protein_classification_t.csv 12 days ago by Sungjae Lee first_submit		0.397	<input type="checkbox"/>
					No more submissions to show			

결론

이번 프로젝트를 통해 가장 크게 느낀 점은, 기계학습 분야가 끊임없는 튜닝과정을 통해 좋은 결과를 낼 수 있는 분야라는 점이었습니다. 한 번에 좋은 결과가 나오는 머신러닝 / 딥러닝 모델은 존재하지 않으며, 주어진 Resource 를 최대한으로 활용하여 과업을 해결하기 위해 다방면으로 도전하고, 최적화해 나가는 것이 핵심이라고 생각하였습니다.

이번 과제에서는 Pretrained 된 모델을 Transfer Learning 을 활용하여 짧은 시간 내에 문제를 해결하였지만, 기회가 된다면 Scratch 부터 학습하였을 때 어느 정도의 성능이 나오는지 연구해보고 싶습니다. 뿐만 아니라 다양한 규제 기법을 제거하거나 다른 것으로 대체하였을 때 성능의 변화가 어느 정도인지 알아보는 과정이 있다면 더 좋은 결과를 얻을 수 있을 것으로 보입니다.