최종 정확도: 76.39222717285156 /총 학습 횟수: 209 /작성자: 201810909 김부용

TRIAL #01

날짜	~2021월 4월 24일					
환경	AWS, pythorch2.7					
정확도 결과	**74.95999908447266**					
중점사항	실습에서 배운 기법을 임의로 적용해서 무조건 많이 학습시켜본다.					
시도 기법	weight initialization	Batch Normalization	Drop out	Learning rate	activation function	epoch
	He Initialization	batch size=16 mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)	0.5	Learning rate decay 20마다 Adam optimizer, learning rate = 0.01	ReLU	87
소요 시간	7 days					
의견	일정 epoch 이후부터 정확도가 오르지 않는 것을 보아 overfitting이 우려된다.					

TRIAL #02

날짜	2021월 4월 25일					
환경	Google colab, pytorch3, GPU 가속					
정확도 결과	**70.4800033569336**					
중점사항	overfitting을 막기 위해 변형한 데이터로 학습시켜본다.					
시도 기법	Data Augmentation	Batch Normalization	Drop out	Learning rate	activation function	epoch
	중앙 확대 + 랜덤 좌우반전	batch size=32 mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)	0.5	Adam optimizer, learning rate = 0.001	ELU	44
소요 시간	16 hours					
의견	한 패턴에 의해 변형된 데이터로만 학습을 너무 많이 시킨 듯하다.					

TRIAL #03

날짜	2021월 4월 26일 ~ 2021월 4월 27일
환경	Google colab, pytorch3
정확도 결과	**70.79000091552734**

<table>
<tr><td>중점사항</td><td colspan="6">데이터를 다양하게 많이 변형해서 수를 늘린 뒤 학습시켜본다.</td></tr>
<tr><td rowspan="10">시도 기법</td><td>Data Augmentation</td><td>Batch Normalization</td><td>Drop out</td><td>Learning rate</td><td>activation function</td><td>epoch</td></tr>
<tr><td>변형x</td><td>batch size=64</td><td rowspan="9">0.5</td><td rowspan="9">Adam optimizer, learning rate = 0.002</td><td rowspan="9">ELU</td><td rowspan="9">42</td></tr>
<tr><td>밝기 80~130%</td><td rowspan="9">mean= (0.5,0.5,0.5), std= (0.5,0.5,0.5)</td></tr>
<tr><td>채도 20~300%</td></tr>
<tr><td>명암 변형</td></tr>
<tr><td>랜덤 크기 변형</td></tr>
<tr><td>중앙 확대</td></tr>
<tr><td>좌우반전</td></tr>
<tr><td>평행성보존 변형</td></tr>
<tr><td>랜덤 시점 변형</td></tr>
<tr><td>소요 시간</td><td colspan="6">2 days</td></tr>
<tr><td>의견</td><td colspan="6">레이어 깊이에 비해 drop out을 너무 많이 적용한 건 아닌가 하는 생각이 든다. 0.5씩이나 적용하지 않으면 정확도가 오히려 좋아질지도 모른다.
그리고 기존에는 변형하지 않은 데이터를 가장 처음 순서로 학습시켰지만 생각해보니 학습이 진행될수록 점점 정상적이지 않은 이미지를 받으면 모델이 혼란스러워할지도 모른다. 다음 시행에서는 변형을 하지 않은 데이터를 마지막 순서로 두어봐야겠다.</td></tr>
</table>

TRIAL #04

<table>
<tr><td>날짜</td><td colspan="6">2021월 5월 3일</td></tr>
<tr><td>환경</td><td colspan="6">Google colab, pytorch3</td></tr>
<tr><td>최대 정확도</td><td colspan="6">73.37999725341797</td></tr>
<tr><td>중점사항</td><td colspan="6">데이터를 다양하게 많이 변형해서 수를 늘리고 Dropout 비율을 조절한다.</td></tr>
<tr><td rowspan="13">시도 기법</td><td>Data Augmentation</td><td>Batch Normalization</td><td>Drop out</td><td>Learning rate</td><td>activation function</td><td>epoch</td></tr>
<tr><td>랜덤 시점 변형</td><td>batch size=64</td><td rowspan="12">차례대로 0.2, 0.3, 0.4, 0.5</td><td rowspan="12">Adam optimizer, learning rate = 0.002</td><td rowspan="12">ELU</td><td rowspan="12">15</td></tr>
<tr><td>랜덤 시점 변형2</td><td rowspan="12">mean= (0.5,0.5,0.5), std= (0.5,0.5,0.5)</td></tr>
<tr><td>평행성보존 변형 (10)+좌우반전</td></tr>
<tr><td>평행성보존 변형 (20)+좌우반전</td></tr>
<tr><td>랜덤 시점 변형 +좌우반전</td></tr>
<tr><td>평행성보존 변형(20)</td></tr>
<tr><td>평행성보존 변형(30)</td></tr>
<tr><td>중앙 확대</td></tr>
<tr><td>평행성보존 변형(10)</td></tr>
<tr><td>평행성보존 변형</td></tr>
</table>

	(30)+좌우반전					
	랜덤 시점 변형2 +좌우반전					
	좌우반전					
	변형x					
소요 시간	10 hours					
의견	변형한 데이터만 너무 학습을 많이 시켜서 정상적인 이미지에 대한 인식이 잘 안 된 걸지도 모른다. 변형하지 않은 데이터를 조금 더 학습시켜 봐야겠다.					

TRIAL #05

날짜	2021월 5월 4일					
환경	Google colab, pytorch3					
최대 정확도	**76.39222717285156**					
중점사항	다양하게 많이 변형한 데이터를 사용해 학습시키되, 변형하지 않은 데이터의 학습을 변형한 데이터보다 한 번 더 시킨다. Dropout 비율을 조절하고 epoch마다 모델을 저장해서 그 중 가장 좋은 정확도가 나온 것을 사용한다.					
시도 기법	Data Augmentation	Batch Normalization	Drop out	Learning rate	activation function	epoch
	랜덤 시점 변형	batch size=64	차례대로 0.1, 0.2, 0.3, 0.5	Adam optimizer, learning rate = 0.002	ELU	21
	랜덤 시점 변형2	mean= (0.5,0.5,0.5), std= (0.5,0.5,0.5)				
	평행성보존 변형 (10)+좌우반전					
	평행성보존 변형 (20)+좌우반전					
	변형x					
	랜덤 시점 변형 +좌우반전					
	평행성보존 변형(20)					
	평행성보존 변형(30)					
	중앙 확대					
	평행성보존 변형(10)					
	평행성보존 변형 (30)+좌우반전					
	랜덤 시점 변형2 +좌우반전					
	좌우반전					
	변형x					
소요 시간	5 hours					
의견	14번째로 저장된 모델에서 가장 좋은 정확도가 나왔으니 이를 최종으로 한다.					