인공지능 팀 프로젝트 보고서
월요일 수업

팀명 : 권전정
팀원 : 201611782 김소연
201611769 강지수
201611845 임소현
201611855 정혜윤

■ 모델 특징.

1. CNN을 이용하여 지역적인 특징을 추출하였다.
2. dropout을 이용하여 학습에 제한을 두어 과적합을 피하였다.
3. MaxPooling2D를 이용하여 레이어의 크기를 줄여 과적합을 피하였다.
4. 여러 배치 사이즈를 돌려 최적의 사이즈를 구하였다.
5. 학습의 반복 횟수를 최대한으로 늘려 정확도를 향상시켰다.

수정하는 값	배치 사이즈	epochs	학습률	정확도
배치 사이즈	4	10	0.1	70.78
	8	10	0.1	73.18
	16	10	0.1	70.86
	32	10	0.1	65.20
	64	10	0.1	60.65
	100	10	0.1	58.25
epochs	8	5	0.1	66.81
	8	10	0.1	73.18
	8	15	0.1	72.48
	8	20	0.1	73.28
학습률	8	20	0.01	63.33
	8	20	0.05	75.23
	8	20	0.1	73.28
	8	20	0.2	nan

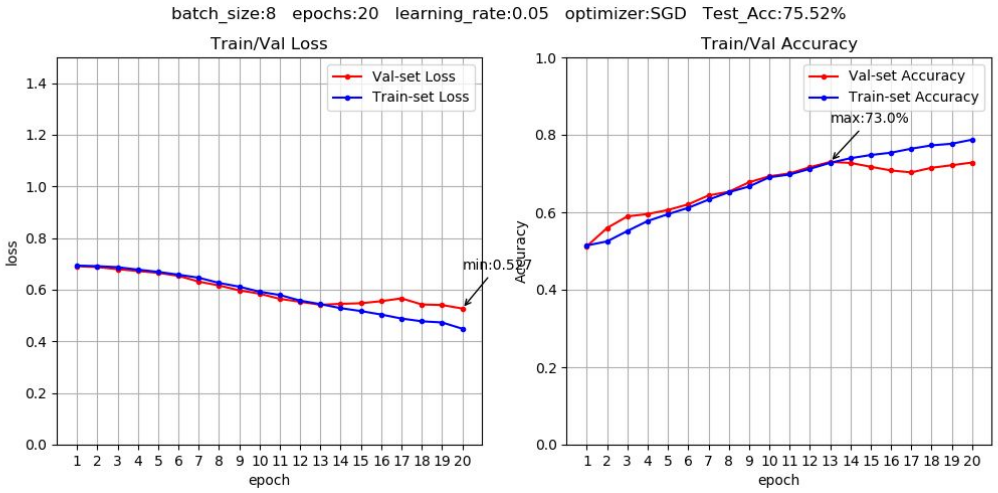
6. Activation을 ELU로 설정하여 0, 1으로 떨어지는 것이 아닌 부드러운 곡선을 그리게 설정하여 정확도를 향상시켰다.

■ 모델 학습 및 실험 결과

1) 최종 모델 성능표 (test data 실험 성능)

지표		값
Accuracy	Total	75.52%
	Cat	71.85%
	Dog	79.20%
Loss		0.5036
FPS		926.96
Model Size		1400KB

2) 학습 그래프 (result.png)



3) 비교 실험

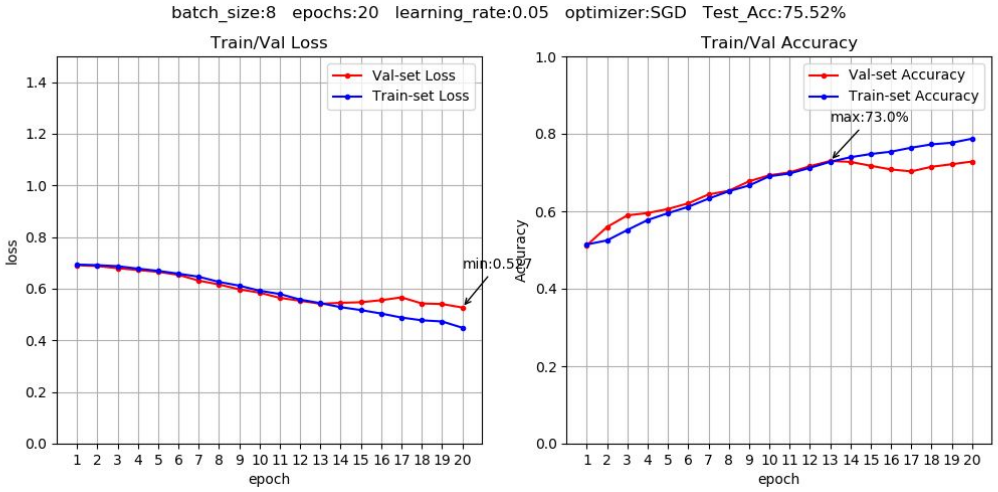
dropout	Train Data	Accuracy Validation Data	Test Data
미사용	8000	70.80	2000
사용	8000	75.52	2000

4) 결과 평가

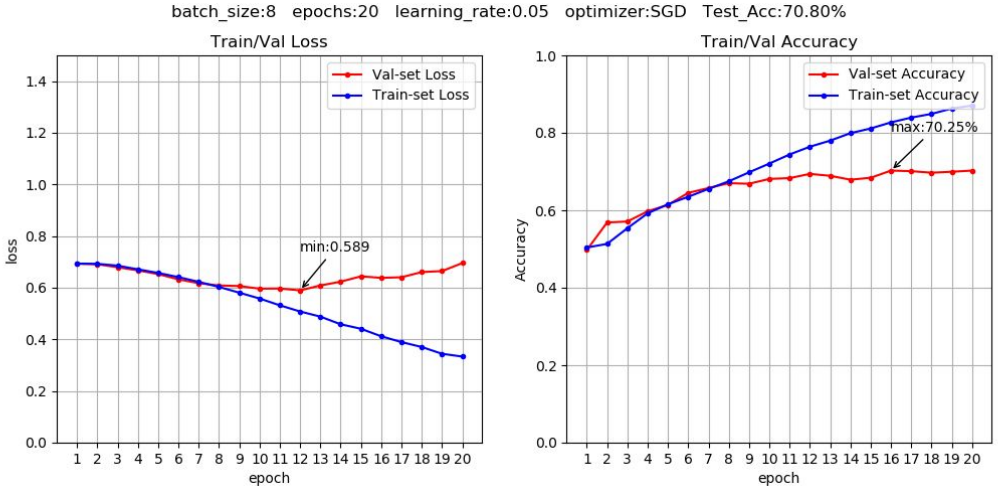
최종 모델 성능의 경우 파라미터 값을 변경하여 실행 하였을 때 가장 성능이 높았던 것에 대한 최종 결과를 기입하였다.

> dropout의 사용여부를 조건으로 비교 실험을 실행한 결과

- dropout 사용시



- dropout 미사용 시



두 가지 모델의 결과를 비교해 본 결과 dropout을 사용한 모델의 최종 정확도 뿐만 아니라 개와 고양이의 정확도가 더 높을 뿐만 아니라 loss값과 FPS값이 더 낮은 것으로 보아 dropout을 사용하는 것이 정확도 측면에 있어 더 효과적이다.

■ 개선될 수 있는 사항

1. 하드웨어 성능 때문에 레이어를 더 추가하지 못했는데, Dense 레이어를 더 추가하면 모델의 정확도 성능이 높아질 것 같다.
2. 현재는 색상 정보가 없는 이미지를 사용하는데, 색상 정보가 추가된다면 더 높은 성능을 얻을 수 있을 것 같다.
3. 데이터가 지금보다 많았으면 해당 데이터 값을 두 그룹으로 나누어 이미 학습 시킨 모델에 재학습을 시키는 방법을 이용하여 정확도를 비약적으로 높일 수 있을 것 같다.
4. hyper['epochs']의 값을 키우면 더 많은 반복 학습을 통하여 모델의 정확도 성능을 높일 수 있을 것 같다.
5. BatchNormalization 사용 가능하다면 일반화를 통하여 모델의 정확도 향상에 도움을 줄 수 있을 것 같다.
6. keras.initializers 모듈의 RandomUniform를 통해 weight초기화를 조절할 수 있다면 loss를 줄여 accuracy를 높일 수 있을 것 같다.

■ 핵심 코드 (get_hyperparameter(), get_model())

get_hyperparameter()

```
hyper['batch_size'] = 8  # 배치 사이즈
hyper['epochs'] = 20  # epochs은 최대 20 설정 !!
hyper['learning_rate'] = 0.05  # 학습률
# 최적화 알고리즘 선택 [sgd, rmsprop, adagrad, adam 등]
hyper['optimizer'] = optimizers.sgd(lr=hyper['learning_rate'])  # default: SGD
```

get_model()

```
model.add(Conv2D(64, (3, 3), padding="same", input_shape=self.x_train.shape[1:], activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(1, activation="sigmoid"))

model.compile(loss="binary_crossentropy", optimizer="rmsprop", metrics=['accuracy'])

model.add(Dense(len(self.target_class)))  # output 예측. (이 부분은 필수)
model.add(Activation('softmax'))  # softmax 함수 적용 (이 부분은 필수)

return model
```