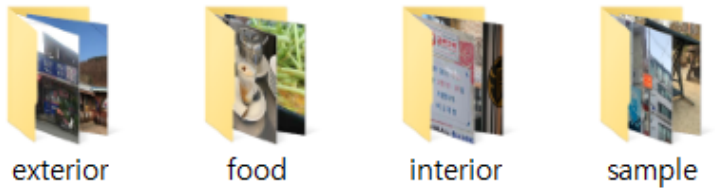
기말 프로젝트

CNN 에 기반한 image classification 알고리즘 개발

201711809 박수빈

45000개의 이미지를 numpy array로 변환



Images폴더에 들어있던 음식, 실내, 실외사진을 폴더를 각각 만들어서 분류했습니다.

그리고 모델 학습 후 예측할 sample사진 10개도 폴더를 만들어서 저장했습니다.

Cv2의 imread함수를 사용해서 이미지파일을 읽어 리스트에 저장했습니다.

Y 배열에는 음식0, 실내1, 실외2로 저장했습니다.

모든 파일을 읽어서 리스트에 추가했으면 np.array로 형식을 변환해줍니다.

```
def train_mnist_model():
    X, y = [], []
    for i in range(20000):
        X.append(cv2.imread('images/food/food%d.jpg'%(i+1)))
        y.append(0)
    print('food insert finished')

    for i in range(15000):
        X.append(cv2.imread('images/interior/interior%d.jpg'%(i+1)))
        y.append(1)
    print('interior insert finished')

    for i in range(10000):
        X.append(cv2.imread('images/exterior/exterior%d.jpg'%(i+1)))
        y.append(2)
    print('exterior insert finished')

    X = np.array(X)
    y = np.array(y)
```

train/test 데이터 구분 및 셔플

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

X와 y는 총 45000개의 data가 들어있고 학습과 테스트모델을 분류하고 데이터가 골고루 들어가기 위해서 train_test_split함수를 사용해 shuffle 했습니다.

나눠진 y_test와 y_train 배열은 0, 1, 2로 이루어져있는데 원핫인코딩하기 위해 to_categorical함수를 사용했습니다.

train data 및 validation data의 loss/accuracy분석

```
def plot_loss_curve(history):

    plt.figure(figsize=(15, 10))

    plt.plot(history['loss'])
    plt.plot(history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper right')
    plt.show()

def plot_accuracy_curve(history):

    plt.figure(figsize=(15, 10))

    plt.plot(history['accuracy'])
    plt.plot(history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper right')
    plt.show()
```

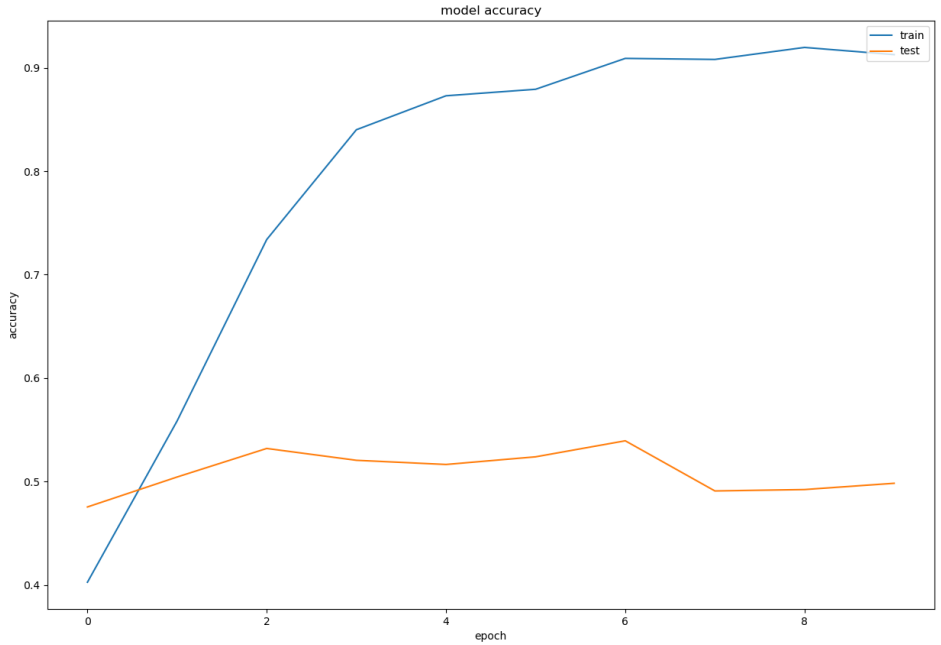
plot_loss_curve는 loss와 val_loss를 epoch가 지나면서 변화를 그래프로 나타내는 함수입니다.

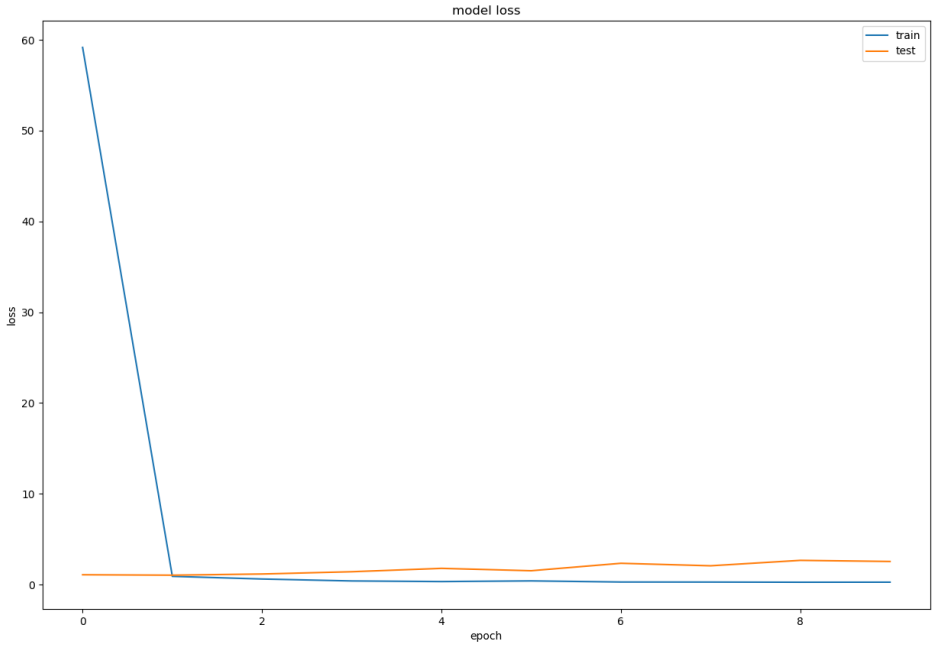
plot_accuracy_curve는 accuracy 와 val_ accuracy를 epoch가 지나면서 변화를 그래프로 나타내는 함수입니다.

train data 및 validation data의 loss/accuracy분석

```
Epoch 1/10
302/302 [==============================] - 351s 1s/step - loss: 277.4598 - accuracy: 0.3888 - val_loss: 1.0864 - val_accuracy: 0.4754
Epoch 2/10
302/302 [==============================] - 394s 1s/step - loss: 0.9163 - accuracy: 0.5688 - val_loss: 1.0391 - val_accuracy: 0.5044
Epoch 3/10
302/302 [==============================] - 352s 1s/step - loss: 0.6052 - accuracy: 0.7364 - val_loss: 1.1677 - val_accuracy: 0.5320
Epoch 4/10
302/302 [==============================] - 358s 1s/step - loss: 0.3732 - accuracy: 0.8501 - val_loss: 1.4186 - val_accuracy: 0.5205
Epoch 5/10
302/302 [==============================] - 337s 1s/step - loss: 0.2896 - accuracy: 0.8896 - val_loss: 1.7894 - val_accuracy: 0.5165
Epoch 6/10
302/302 [==============================] - 332s 1s/step - loss: 0.3277 - accuracy: 0.9082 - val_loss: 1.5286 - val_accuracy: 0.5239
Epoch 7/10
302/302 [==============================] - 326s 1s/step - loss: 0.2728 - accuracy: 0.9157 - val_loss: 2.3479 - val_accuracy: 0.5394
Epoch 8/10
302/302 [==============================] - 346s 1s/step - loss: 0.2481 - accuracy: 0.9238 - val_loss: 2.0734 - val_accuracy: 0.4909
Epoch 9/10
302/302 [==============================] - 345s 1s/step - loss: 0.2454 - accuracy: 0.9230 - val_loss: 2.6707 - val_accuracy: 0.4923
Epoch 10/10
302/302 [==============================] - 346s 1s/step - loss: 0.2731 - accuracy: 0.9146 - val_loss: 2.5483 - val_accuracy: 0.4983
```

train data 및 validation data의 loss/accuracy분석





Model 구성

```
model = Sequential([
            Input(shape=(300,300,3), name='input_layer'),
            Conv2D(32, kernel_size=3, activation='relu', name='conv_layer1'),
            MaxPooling2D(pool_size=2),
            Conv2D(64, kernel_size=3, activation='relu', name='conv_layer2'),
            MaxPooling2D(pool_size=2),
            Dropout(0.5),
            Flatten(),
            Dense(64, activation='relu', name='output_layer1'),
            Dense(3, activation='softmax', name='output_layer2')
        ])
```

Layer을 추가하면 파라미터 개수가 너무 많아져서 시간이 많이걸려서 layer의 개수를 몇개할지 고민하다가 2개로 결정했습니다.

Conv2d 3*3필터를 2개 사용했고 활성화함수로 relu함수를 사용했고 사진을 너무 자세히 보지 않고 크게보게하기 위해 maxpooling2d필터를 사용했습니다.

오버피팅을 막기 위해서 dropout함수를 사용했습니다.

Model 구성

Layer (type)	Output Shape	Param #
conv_layer1 (Conv2D)	(None, 298, 298, 32)	896
max_pooling2d (MaxPooling2D)	(None, 149, 149, 32)	0
conv_layer2 (Conv2D)	(None, 147, 147, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 73, 73, 64)	0
dropout (Dropout)	(None, 73, 73, 64)	0
flatten (Flatten)	(None, 341056)	0
output_layer1 (Dense)	(None, 64)	21827648
output_layer2 (Dense)	(None, 3)	195

Total params: 21,847,235
Trainable params: 21,847,235
Non-trainable params: 0

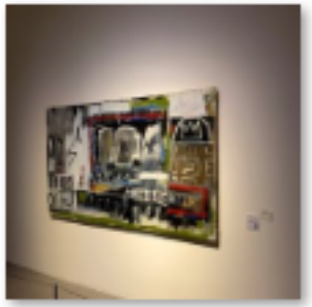
Prediction 시연

sample1

sample2

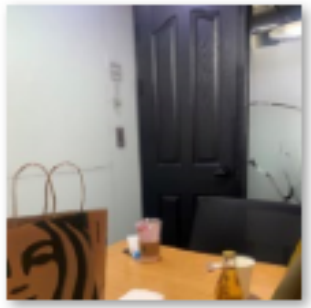
sample3

sample4



sample5

sample6



sample7

sample8

sample9

sample10

Sample사진들은 제가 직접 찍은 사진들로 음식사진4개 실내사진3개 실외사진 3개로 총 10개로 구성했습니다.

Prediction 시연

Prediction 시연

성능을 높이기 위해 노력한 방법들

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_layer1 (Conv2D)         (None, 298, 298, 32)      896
_________________________________________________________________
dropout (Dropout)            (None, 298, 298, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 32)      0
_________________________________________________________________
flatten (Flatten)            (None, 710432)            0
_________________________________________________________________
output_layer (Dense)         (None, 3)                 2131299
=================================================================
Total params: 2,132,195
Trainable params: 2,132,195
Non-trainable params: 0
_________________________________________________________________
2020-12-17 11:19:29.268000: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
Epoch 1/10
302/302 [==============================] - 277s 917ms/step - loss: 2160.0946 - accuracy: 0.4766 - val_loss: 9.3333 - val_accuracy: 0.5199
Epoch 2/10
302/302 [==============================] - 264s 874ms/step - loss: 2.4004 - accuracy: 0.8522 - val_loss: 7.9187 - val_accuracy: 0.5859
Epoch 3/10
302/302 [==============================] - 267s 884ms/step - loss: 0.4653 - accuracy: 0.9460 - val_loss: 7.9620 - val_accuracy: 0.5879
Epoch 4/10
302/302 [==============================] - 269s 890ms/step - loss: 0.1566 - accuracy: 0.9790 - val_loss: 9.3590 - val_accuracy: 0.5906
Epoch 5/10
302/302 [==============================] - 261s 864ms/step - loss: 0.2129 - accuracy: 0.9780 - val_loss: 8.7897 - val_accuracy: 0.5690
Epoch 6/10
302/302 [==============================] - 246s 814ms/step - loss: 0.1405 - accuracy: 0.9796 - val_loss: 10.2960 - val_accuracy: 0.5859
Epoch 7/10
302/302 [==============================] - 235s 778ms/step - loss: 0.0562 - accuracy: 0.9880 - val_loss: 8.9190 - val_accuracy: 0.5778
Epoch 8/10
302/302 [==============================] - 236s 780ms/step - loss: 0.1313 - accuracy: 0.9877 - val_loss: 8.6333 - val_accuracy: 0.5609
Epoch 9/10
302/302 [==============================] - 299s 991ms/step - loss: 0.1966 - accuracy: 0.9799 - val_loss: 11.4515 - val_accuracy: 0.5623
Epoch 10/10
302/302 [==============================] - 235s 778ms/step - loss: 0.5420 - accuracy: 0.9574 - val_loss: 15.3874 - val_accuracy: 0.5805
```

모델의 layer구성도 다르게 해보고 epoch수와 batch사이즈도 변경하면서 테스트했는데 정확도는 거의 50퍼로 비슷하게 나왔습니다.

사진이 45000이기 때문에 시간이 너무 오래 걸려서 4500장으로 추려서 테스트를 많이 해봤습니다.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_layer1 (Conv2D)         (None, 298, 298, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 32)      0
_________________________________________________________________
flatten (Flatten)            (None, 710432)            0
_________________________________________________________________
output_layer (Dense)         (None, 3)                 2131299
=================================================================
Total params: 2,132,195
Trainable params: 2,132,195
Non-trainable params: 0
_________________________________________________________________
2020-12-17 05:17:25.857508: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
Epoch 1/3
3015/3015 [==============================] - 1403s 465ms/step - loss: 283.8305 - accuracy: 0.5232 - val_loss: 1.0808 - val_accuracy: 0.5244
Epoch 2/3
3015/3015 [==============================] - 1443s 479ms/step - loss: 0.5292 - accuracy: 0.8020 - val_loss: 1.6497 - val_accuracy: 0.5370
Epoch 3/3
3015/3015 [==============================] - 1460s 484ms/step - loss: 0.4652 - accuracy: 0.8562 - val_loss: 1.7208 - val_accuracy: 0.4914
{'loss': [48.97663497924805, 0.5803033113479614, 0.5584049224853516], 'accuracy': [0.5273631811141968, 0.7763184309005737, 0.8064345121383667], 'val_loss': [1.0808273553848267, 1.649685025215149, 1.7208101749420166], 'val_accuracy': [0.5243771076202393, 0.5370370149612427, 0.4914478063583374]}
train loss= 0.5584049224853516
validation loss= 1.7208101749420166
2020-12-17 06:31:44.278592: W tensorflow/python/util/util.cc:348] Sets are not currently considered sequences, but this may change in the future, so consider avoiding using them.
```

감사합니다