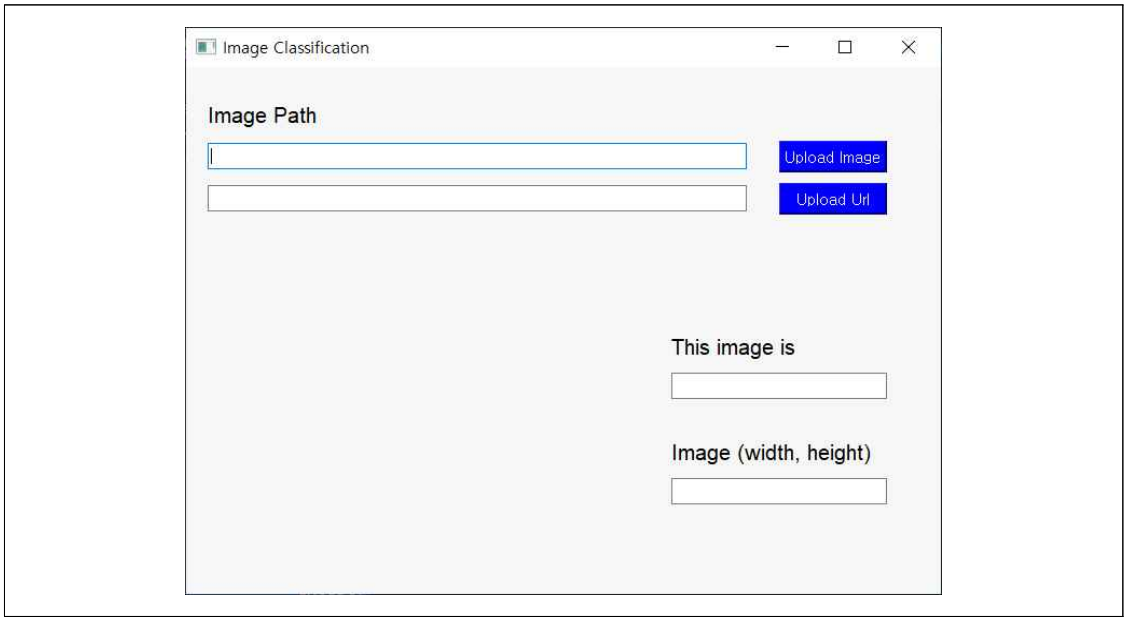
과목명	딥러닝/클라우드
담당교수	오세종 교수님
학과	소프트웨어학과
학번	32153180
이름	이상민
제출일자	2020.12.10

■ 과제 내용

- 이미지 파일을 업로드하면 CIFAR-10 의 레이블 중 하나를 예측하여 보여주는 소프트웨어를 개발한다.
- 예측 모델은 전이학습을 이용하여 만들고 CIFAR-10을 이용하여 학습한다.
- UI는 다음과 같은 형태

■ 실행 화면

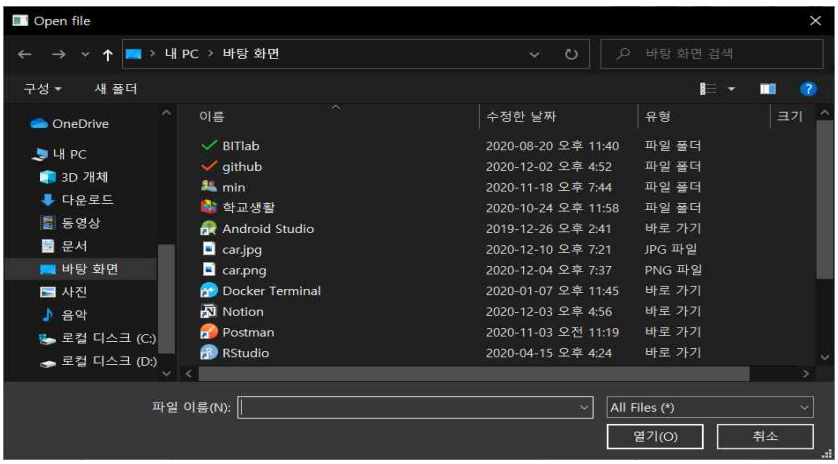
- GUI



- Button

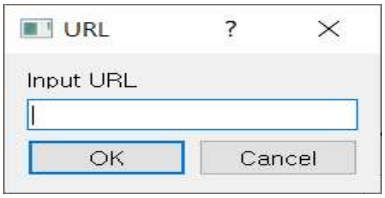
1. *Upload Image*

   pyqt의 QFileDialog 창을 띄워 원하는 경로에 있는 이미지를 선택해서 입력

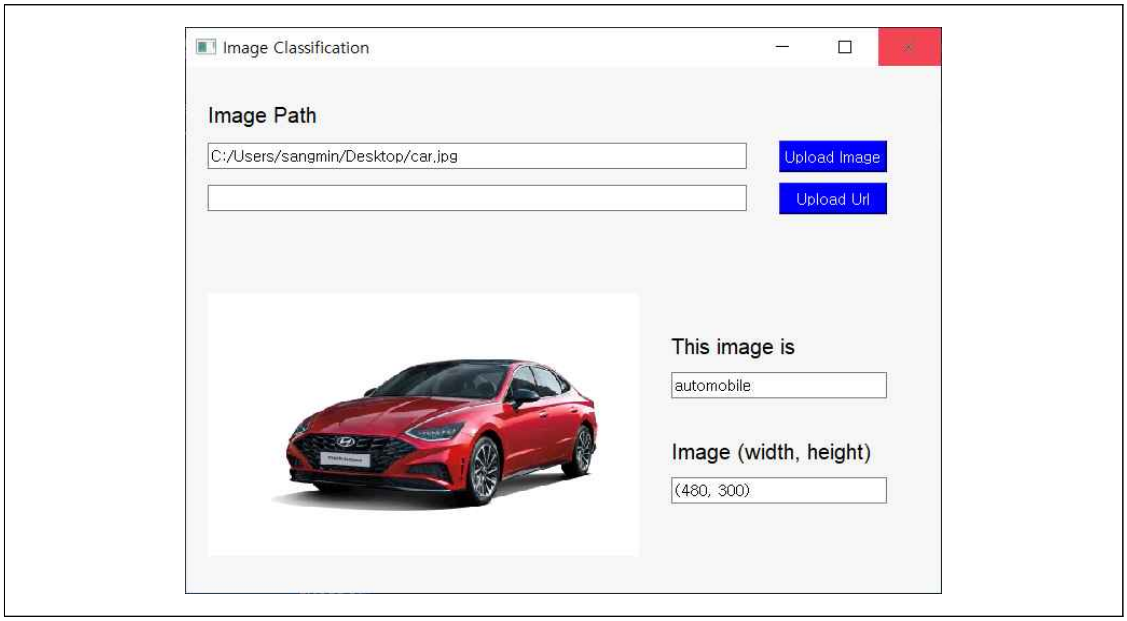


2. *Upload Url*

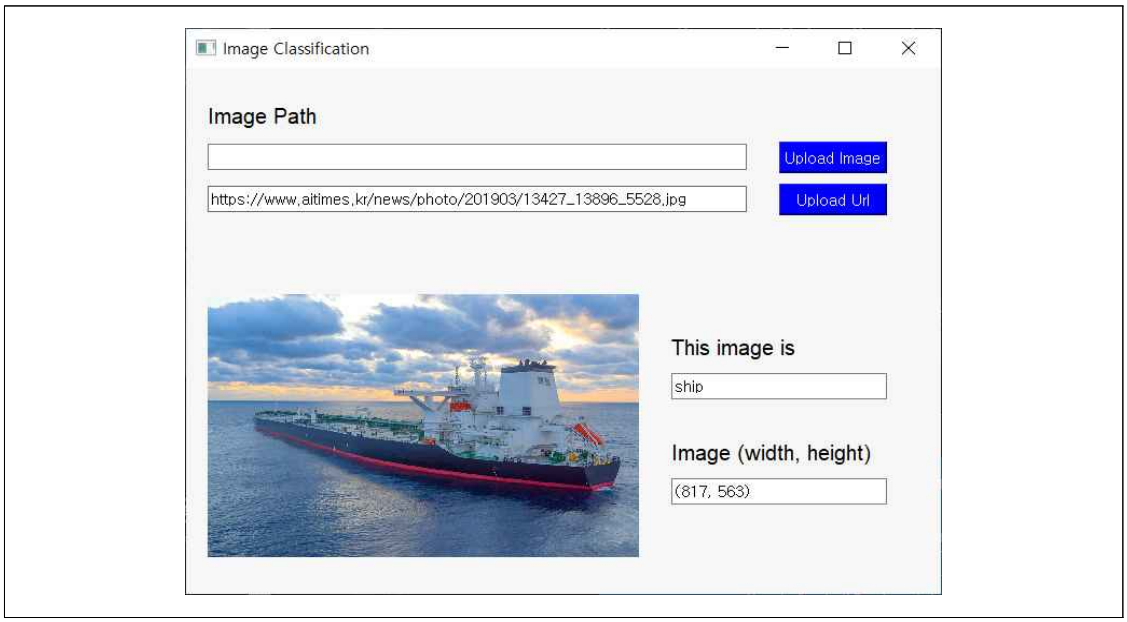
   QInputDialog에 이미지 주소 입력



- Upload Image



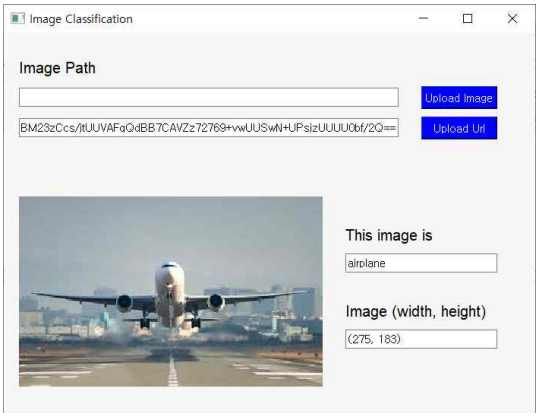
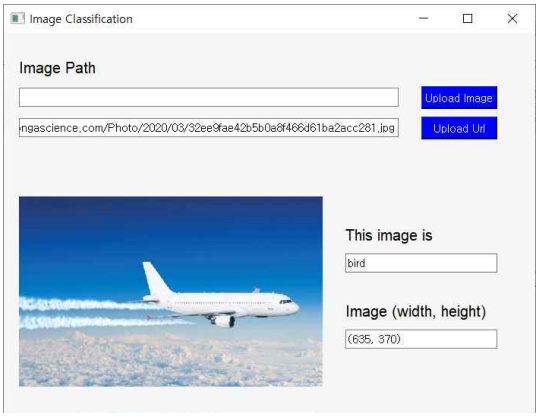
- Upload Url



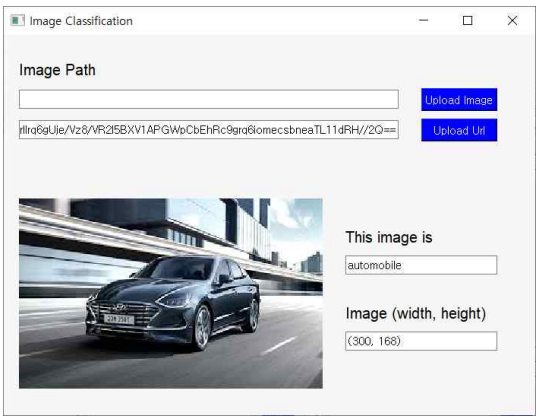
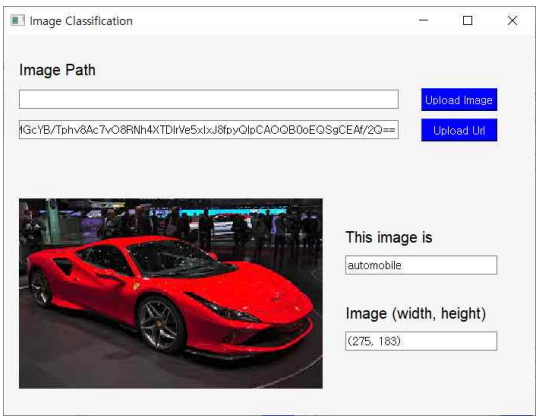
- Information
  - 선택한 이미지의 경로 혹은 URL 표시
  - 이미지 표시
  - 해당 이미지의 분류 예측 결과 표시
  - 원본 이미지 사이즈 표시

■ 테스트

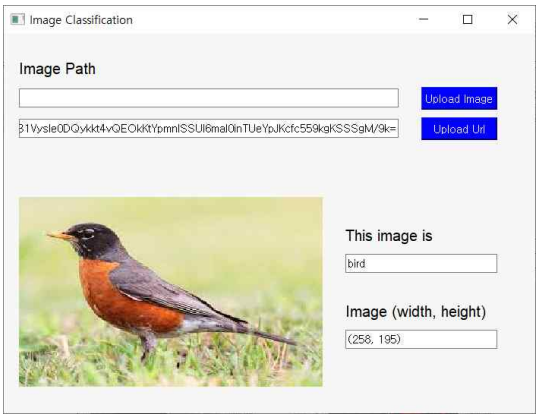
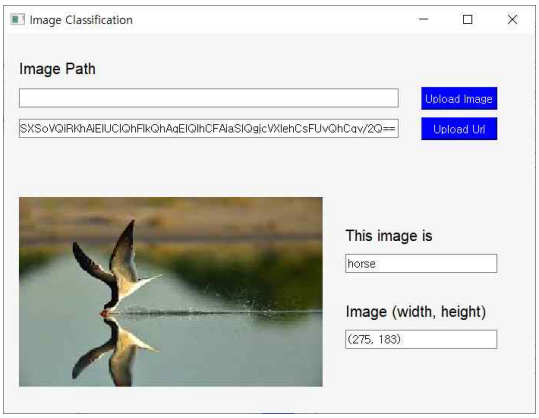
- Airplane

Correct	Incorrect
	

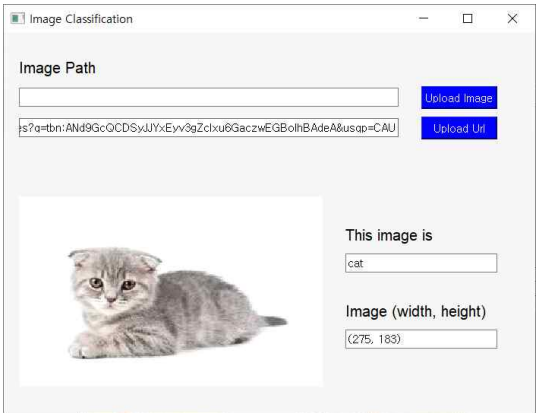
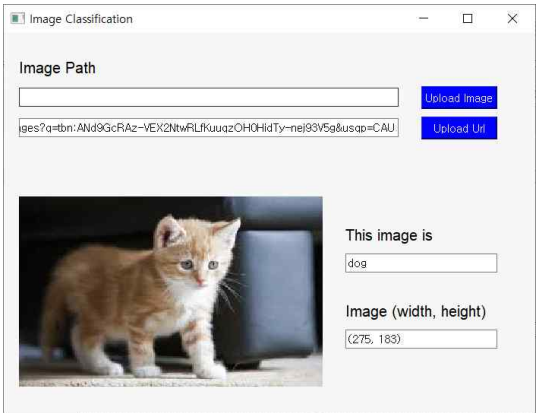
- Automobile

Correct	
	

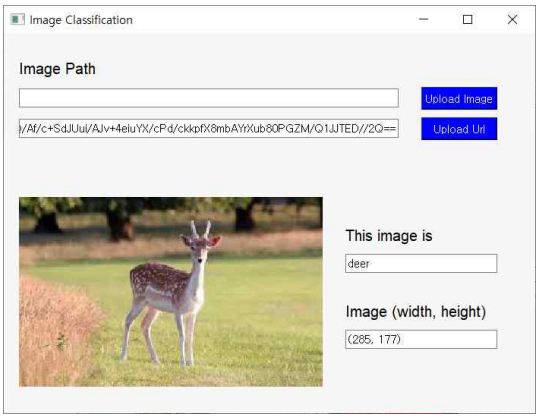
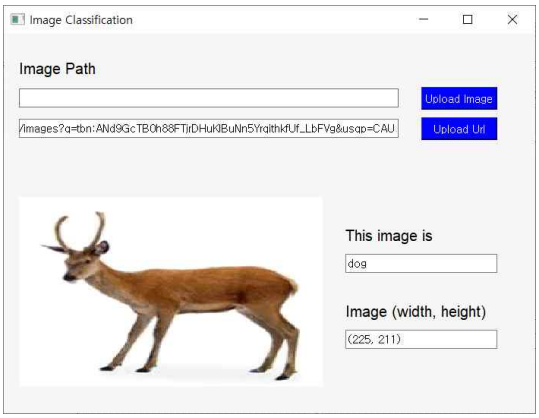
- Bird

Correct	Incorrect
	

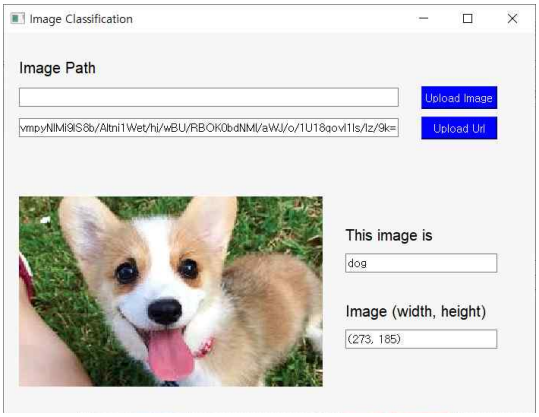
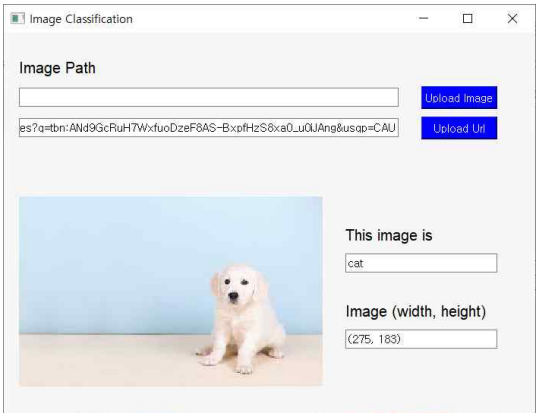
- Cat

Correct	Incorrect
	

- Deer

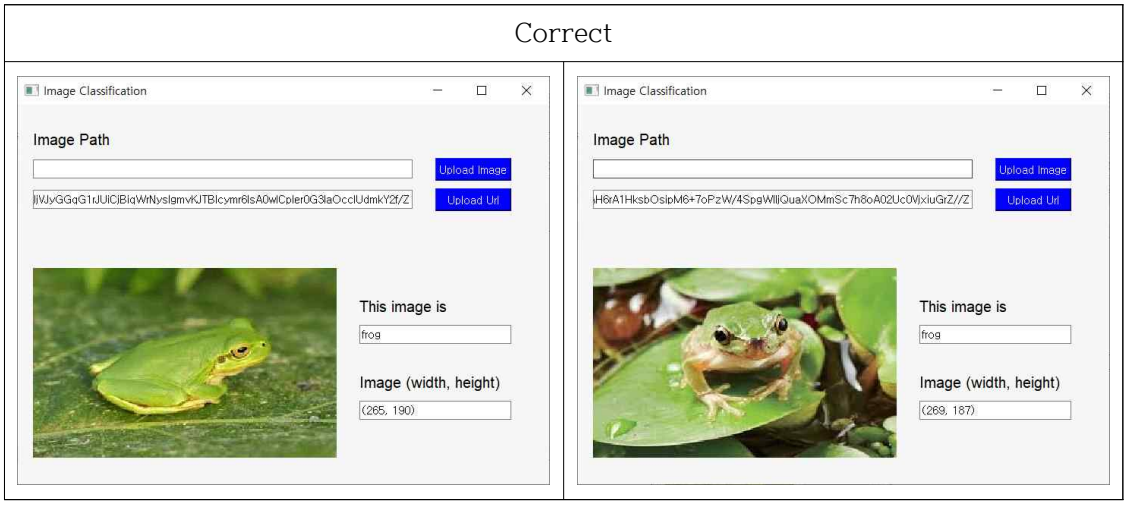
Correct	Incorrect
	

- Dog

Correct	Incorrect
	

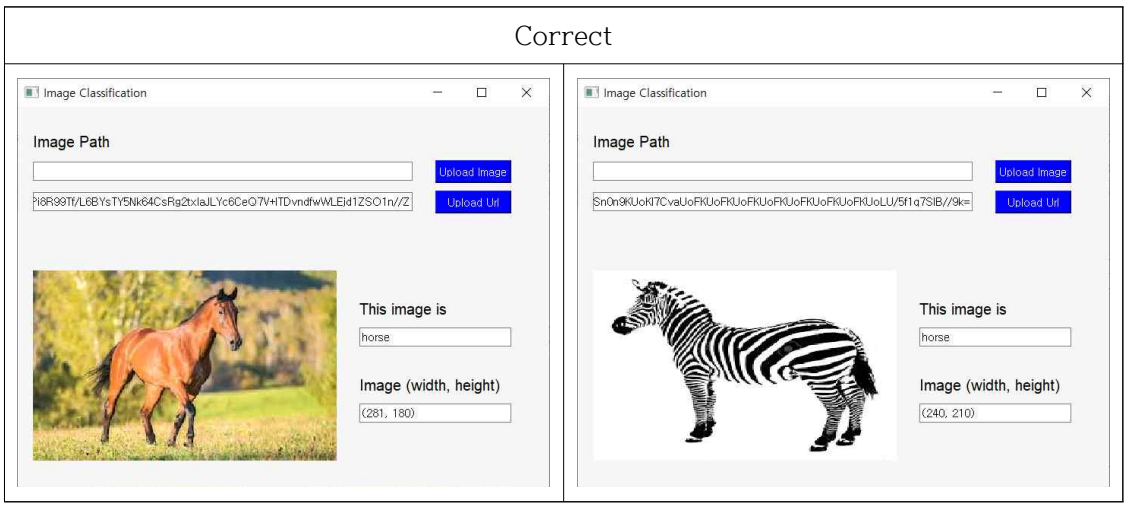
- Frog

Correct

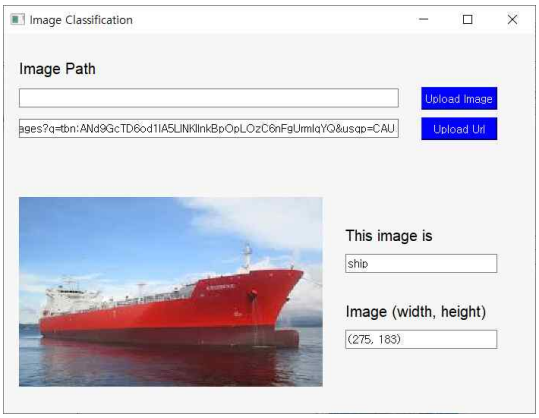
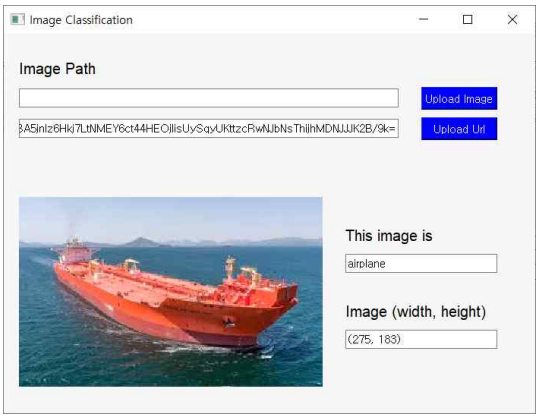


- Horse

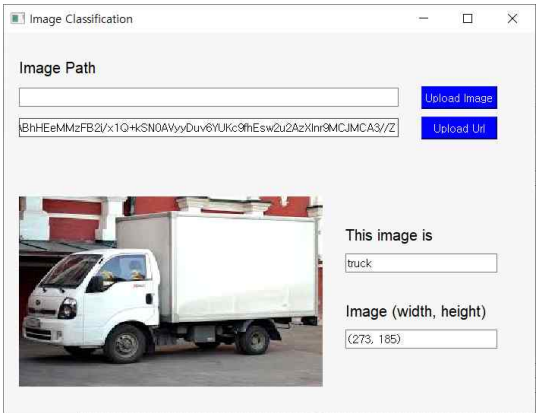
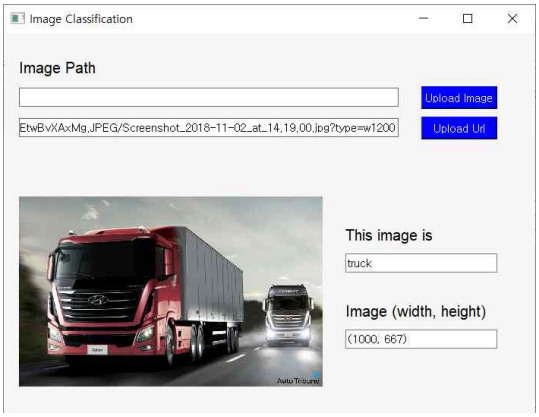
Correct



- Ship

Correct	Incorrect
	

- Truck

Correct	
	

■ 소감

모델도 모델이지만 GUI 만드는 것을 한 번도 해본 적이 없어 생소했다. 최대한 예시로 주어진 GUI와 비슷하게 만들기 위해 노력했다. QFildDialog를 이용해 사진 파일을 불러오는 것은 쉽게 구현할 수 있었다. 하지만 URL로 사진을 받아오는 작업에서 꽤나 애먹었다. URL을 텍스트 자체로 읽어들이는 것은 쉽지만 pyqt 특성 상 Qpixmap 함수를 이용해 이미지로 바꿔주는데, 이렇게 바뀐 이미지를 학습한 모델에 넣으면 계속 이상한 값이 나왔다.

학습 모델은 VGG16 아키텍처를 이용했다. 교수님께서 수업 시간에 VGG의 실행 시간이 다른 것보다 길다고 말씀하셔서 Google colab을 통해 모델을 학습시켰다. 처음엔 모델 학습 시간이 오래 걸릴 것을 대비하여 epoch을 굉장히 낮게 잡았다. 모델 정확도는 높게 나왔지만 이미지를 대입할 경우 제대로 된 예측을 하지 못했다. 그리고 생각한 것 만큼 학습 시간이 길지 않아 epoch을 크게 50으로 설정했다.

앞 장의 테스트에서 제시했던 것처럼 100% 정확한 예측을 하진 않는다. 하지만 몇몇 클래스에 대해서는 상당히 높은 정확도를 보여준다. 사람의 눈은 비행기 사진의 배경이 하늘이라는 것과 선박 사진의 배경이 바다라는 사실을 당연하게 구분하지만, 컴퓨터는 그렇지 않다는 점을 깨달았다. 또 이와 같은 예측 모델의 정확도를 높이기 위해 밤낮으로 연구하는 개발자들이 존경스럽다.

■ 소스 코드

- Model

```
from keras import optimizers
from keras.datasets import cifar10
from keras.engine import Model
from keras.layers import Dropout, Flatten, Dense
from keras.utils import np_utils
from keras.applications.vgg16 import VGG16

img_width, img_height = 32, 32
base_model = VGG16(weights='imagenet',
                   include_top=False,
                   input_shape=(32, 32, 3))

nb_epoch = 50
nb_classes = 10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
y_train = np_utils.to_categorical(y_train, nb_classes)
y_test = np_utils.to_categorical(y_test, nb_classes)

last = base_model.get_layer('block5_pool').output

x = Flatten()(last)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(10, activation='softmax')(x)

model = Model(base_model.input, output)
```

```
model.compile(loss='binary_crossentropy',
            optimizer=optimizers.SGD(lr=1e-3, momentum=0.9),
            metrics=['accuracy'])

model.fit(X_train, y_train,
         validation_data=(X_test, y_test),
         epochs=nb_epoch,
         batch_size=200,
         verbose=1)

scores = model.evaluate(X_test, y_test, verbose=0)
print("loss: %.2f" % scores[0])
print("acc: %.2f" % scores[1])

model.save('cifar10-model.h5')
```

- Image Classification SW

```
import sys
import numpy as np
import urllib.request
import qimage2ndarray
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.models import load_model
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array

from PIL import Image
from urllib import request
from io import BytesIO

class GUI(QWidget):
    def __init__(self):
        super().__init__()

        self.model = load_model('C:/Users/sangmin/Desktop/학교생활/4-2/
딥러닝클라우드/cifar10-model.h5')

        self.setWindowTitle("Image Classification")
        self.setGeometry(100, 100, 700, 500)
        self.initUI()
```

```
    def initUI(self):
        title_label = QLabel('Image Path', self)
        title_label.setGeometry(20, 30, 300, 30)
        title_label.setFont(QFont('Arial', 12))

        info_label = QLabel('Image (width, height)', self)
        info_label.setGeometry(450, 350, 300, 30)
        info_label.setFont(QFont('Arial', 12))

        self.info_line = QLineEdit(self)
        self.info_line.setGeometry(450, 390, 200, 25)

        sub_label = QLabel('This image is', self)
        sub_label.setGeometry(450, 250, 300, 30)
        sub_label.setFont(QFont('Arial', 12))

        self.image_label = QLabel('', self)
        self.image_label.setGeometry(20, 90, 500, 500)

        self.upload_line = QLineEdit(self)
        self.upload_line.setGeometry(20, 72, 500, 25)
        upload_button = QPushButton(self)
        upload_button.setText('Upload Image')
        upload_button.setStyleSheet('color: white; background: blue')
        upload_button.resize(100, 30)
        upload_button.move(550, 70)
        upload_button.clicked.connect(self.upload_file)

        self.url_line = QLineEdit(self)
        self.url_line.setGeometry(20, 112, 500, 25)
        url_button = QPushButton(self)
        url_button.setText('Upload Url')
        url_button.setStyleSheet('color: white; background: blue')
        url_button.resize(100, 30)
        url_button.move(550, 110)
        url_button.clicked.connect(self.upload_url)

        self.class_line = QLineEdit(self)
        self.class_line.setGeometry(450, 290, 200, 25)

        self.show()
```

```
    def upload_file(self):
        file = QFileDialog.getOpenFileName(self, 'Open file')
        self.upload_line.setText(file[0])

        if file[0]:
            pixmap = QPixmap(file[0])
            pixmap = pixmap.scaledToHeight(250)
            self.image_label.setPixmap(pixmap)

            img = load_img(file[0])
            w, h = img.size[0], img.size[1]
            self.info_line.setText("({}, {})".format(w, h))
            image = img.resize((32, 32))
            self.image_classification(image)
        else:
            QMessageBox.about(self, "Warning", "사진을 선택하지 않았습니
다.")
```

```
    def upload_url(self):
        text, status = QInputDialog.getText(self, 'URL', 'Input URL')

        if status:
            try:
                self.url_line.setText(str(text))

                image = urllib.request.urlopen(text).read()
                pixmap = QPixmap()
                pixmap.loadFromData(image)
                pixmap = pixmap.scaled(400, 250)
                self.image_label.setPixmap(pixmap)

                img = urllib.request.urlopen(text)
                img = Image.open(img)
                w, h = img.size[0], img.size[1]
                self.info_line.setText("({}, {})".format(w, h))
                img = img.resize((32, 32))
                self.image_classification(img)

            except:
                QMessageBox.about(self, "Warning", "유효하지 않은 URL입니
다.")

    def image_classification(self, image):
        category = ['airplane', 'automobile', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck']
        image = img_to_array(image)
        image  =  image.reshape((1,  image.shape[0],  image.shape[1],
image.shape[2]))
        image = preprocess_input(image)
        pred = self.model.predict(image)
        pred_class = category[np.argmax(pred)]
        self.class_line.setText(pred_class)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = GUI()
    app.exec_()
```