Major Neural Network Architectures

이번 한 주간 CNN, U-Net, Autoencoder, GAN 등 다양한 주요 신경망 구조들에 대해서 배워봤습니다. 오늘은 그 모델들을 복습하는 시간을 가지도록 하겠습니다. 이 SC는 **신경망의 다양한 구조에 대한 이해와 지식**을 평가합니다. **모델을 높은 정확도를 가지도록 학습 시킬 수 있는지를 평가하려는 것이 아닙니다.**

아래의 방식들은 복잡한 연산을 요구합니다. 모든 파트의 문제들은 어떤 환경에서라도 (e.g. 로컬 주피터, Google Colab, etc.) 5-10분 내외로 결과값이 나오도록 제작이 됐기 때문에 만일 결과값을 도출하는데 그 이상의 시간이 걸린다면 여러분의 접근 방식을 재점검해보시기 바랍니다.

---

1. CNN

객체 탐지 모델

Keras와 ResNet50v2 (pre-trained)을 활용하여 `im_frog` 폴더에 있는 이미지 중 어떤 이미지에 개구리가 있는지 찾는 객체 탐지 모델을 만들어 보겠습니다 (주의: 해당 이미지들을 Colab에 업로드 하셔야 합니다. 폴더이름은 다른 것을 사용하셔도 좋고, 아래 코드에서도 같이 바꾸셔야 합니다.)

2.1 ResnetV_2을 사용하기위해서 전처리 함수를 사용하여 이미지를 전처리 하고 이미지들의 사이즈를 재조정하세요 (Hint: ImageDataGenerator, scikit-image)

In [1]:
```
from google.colab import drive

drive.mount('/content/drive')
```
```
Mounted at /content/drive
```

In [2]:
```
import os

frog_dir = '/content/drive/My Drive/SC43/im_frog'
```

In [3]:
```
filenames = os.listdir(frog_dir)
```

In [10]:
```
from keras.preprocessing.image import load_img, img_to_array

import numpy as np

def load_and_preprocessing(base_dir, name, model):
    """
    이미지 1장을 받아 모델로 예측한 뒤
    가장 확률이 높은 클래스 번호를 출력하는 함수입니다.

    Hint:
        1. 경로에서 이미지를 불러옵니다.
        2. array의 값을 직접 나누어 픽셀 값을 정규화합니다.

    Args:
        base_dir : 이미지 파일이 있는 경로입니다.
        name : 이미지 파일의 이름입니다.
        model : 예측에 사용할 모델입니다.
    """
    image_path = base_dir + '/' + name
    image = load_img(image_path, target_size=(224, 224))
    input_arr = img_to_array(image) / 255.0

    input_arr = np.array([input_arr])
    predictions = model.predict(input_arr)
    predict_class = np.argmax(predictions, axis=1)
    print(predict_class[0])

    return predict_class[0]
```

2.2 ResNet50v2 모델을 사용해 `images`를 분류하세요. 예측 결과는 자유롭게 출력하세요.

> 참고: `ResNet50v2`는 "frog"로 예측하지 않습니다. "frog"의 label은 "bullfrog, treefrog, tailed frog"입니다

In [ ]:
```
from tensorflow.keras.applications.resnet_v2 import ResNet50V2, decode_predictions, preprocess_input

resnetV2 = ResNet50V2(weights='imagenet', include_top = True, input_shape=(224,224,3))
resnetV2.summary()
```

In [12]:
```
predict_class = [load_and_preprocessing(frog_dir, filename, resnetV2) for filename in filenames]
```
```
31
58
397
31
738
985
32
58
30
807
309
985
868
308
113
```

In [13]:
```
!git clone https://github.com/anishathalye/imagenet-simple-labels.git
```
```
Cloning into 'imagenet-simple-labels'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 21 (delta 1), reused 10 (delta 1), pack-reused 10
Unpacking objects: 100% (21/21), done.
```

In [14]:
```
import json

with open ('/content/imagenet-simple-labels/imagenet-simple-labels.json') as f:
    labels = json.load(f)

for i in predict_class:
    print(labels[i])
```
```
tree frog
water snake
pufferfish
tree frog
pot
daisy
tailed frog
water snake
American bullfrog
solar thermal collector
bee
daisy
tray
fly
snail
```

---

2. U-Net

Lecture Note에서는 U-Net의 백본(backbone)모델로 `MobileNetV2`를 사용하여 segmentation을 수행하였습니다.
이번 SC에서는 ResNet50을 백본으로 하여 같은 문제를 풀어보세요.

참고로 resnet에서의 block은 아래 예시의 3개의 레이어를 참조하여 만들어주세요.
예시는 16x16 까지만 나타나 있지만 Lecture Note 와 같이 4x4까지 만들어 주어야 모델을 완성할 수 있습니다.

```
        'conv1_relu', # 64x64
        'conv2_block3_out', # 32x32
        'conv3_block4_out', # 16x16
```

In [15]:
```
# Note 에 구현된 U-Net 모델 정의하기 부분에서
# Downstack 부분을 아래와 같이 구현합니다.
# 나머지 부분은 동일하며 실제 재제출 시에는 나머지 코드도 작성하여 모델을 완성시켜주세요.

base_model = tf.keras.applications.ResNet50(input_shape=[128, 128, 3], include_top=False) # 기존 인풋사이즈 재사용 False

#이 층들의 활성화를 이용합시다
layer_names = [
    'conv1_relu', # 64x64
    'conv2_block3_out', # 32x32
    'conv3_block4_out', # 16x16
    'conv4_block6_out', # 8x8
    'conv5_block3_out', # 4x4
]
layers = [base_model.get_layer(name).output for name in layer_names]

# 특징추출 모델을 만듭시다
down_stack = tf.keras.Model(inputs=base_model.input, outputs=layers)

down_stack.trainable = False
```