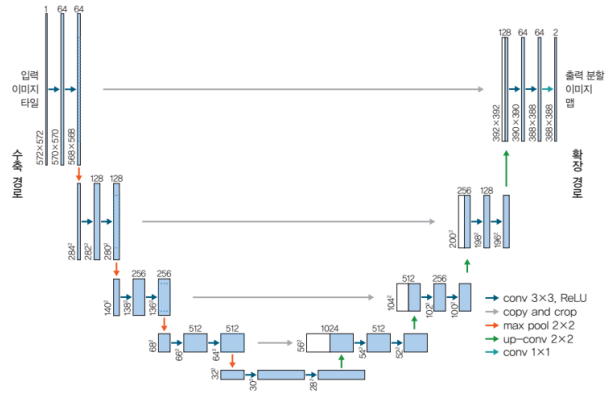
U-Net

- U-Net
  - U-Net은 의료 영상 처리와 같은 고해상도 이미지 영역 분할 작업에 특히 뛰어난 성능을 보이는 딥러닝 아키텍처
  - 이 모델은 특히 세밀한 이미지 영역 분할을 필요로 하는 분야에서 강력한 도구로 자리 잡았음
  - 특히나 U-Net이 제안한 아키텍처는 최신 인공지능 모델에서까지 많은 영향을 주게 되었음
  - U-Net은 주로 의료 이미지 분할을 위해 설계된 특별한 형태의 합성곱 신경망
  - 이 네트워크는 그 구조가 'U' 형태를 닮았다 하여 U-Net이라 명명되었으며, 두 가지 주요 구성 요소인 수축 경로(contracting path)와 확장 경로(expanding path)를 포함

1

U-Net

▼ 그림 5-39 U-Net 구조



2

U-Net

- U-Net

  수축 경로

  - U-Net의 수축 경로는 주로 네트워크의 앞부분에 위치하며, 이는 전통적인 합성곱 신경망의 구조와 유사

▼ 그림 5-40 수축 경로 세부 정보

3

U-Net

- U-Net
  - 이 경로의 주요 목적은 이미지로부터 중요한 특징을 추출하는 것
  - 수축 경로는 여러 합성곱 층과 풀링 층으로 구성되어 있으며, 각 층은 이미지에서 다양한 수준의 특징을 학습하는 데 중요한 역할을 함
    - **합성곱 층**: 수축 경로의 시작 부분에 있는 합성곱 층은 이미지의 원시 픽셀 값에서 복잡한 특징을 추출

      이 층에서 사용되는 필터(또는 커널)는 이미지의 작은 영역에 적용되어 지역적인 특징을 감지

      예를 들어 가장자리, 질감, 패턴 등이 이 단계에서 학습

      합성곱 층을 거치면서 입력 이미지는 여러 특징 맵으로 변환되는데, 이는 각 필터가 이미지의 특정 종류의 특징에 반응하기 때문임

4

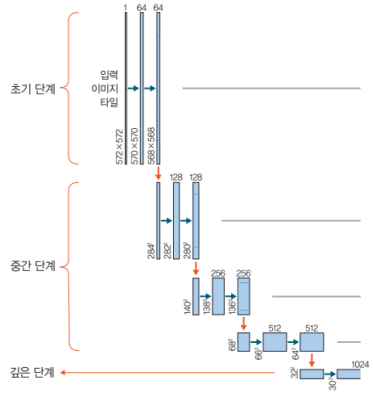
U-Net

- U-Net
  - **폴링 층**: 각 합성곱 층 뒤에는 폴링 층이 따름
    폴링은 주로 최대 폴링 방식을 사용하는데, 이는 각 특징 맵의 지역적인 영역에서 가장 높은 값을 선택하여 그 사이즈를 줄이는 과정
    이렇게 함으로써 폴링 층은 이미지의 차원을 줄이고, 중요한 특징을 유지하면서도 데이터의 양을 감소시킴
    이 과정은 네트워크가 더 넓은 범위의 맥락 정보를 학습하도록 하며, 과적합을 방지하는 데도 도움이 됨

5

U-Net

▼ 그림 5-41 수축 경로 아키텍처



6

U-Net

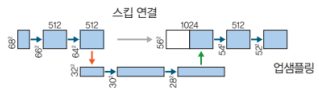
- U-Net
  - **초기 단계**: 이 단계는 이미지의 가장 기본적인 특징을 감지
    여기에는 주로 간단한 패턴, 가장자리, 색상 변화 등이 포함
    이 초기 층은 이미지의 원시 픽셀 데이터에서 직접 정보를 추출하기 때문에 비교적 단순한 특징에 집중
  - **중간 단계**: 수축 경로를 따라 내려가면서 네트워크는 점점 더 복잡한 특징을 학습
    이 단계에서는 텍스처, 패턴의 조합, 그리고 이미지의 일부 형태나 구조 같은 좀 더 복잡한 요소들이 감지
    이러한 중간 스테이지는 이미지의 고유한 특성을 더 잘 이해하고 구별하는 데 도움을 줌
  - **깊은 단계**: 네트워크의 더 깊은 부분에서는 이미지의 고수준 특징이 처리
    여기에는 객체의 큰 형태나 전체적인 구조 같은 더 복잡하고 추상적인 정보가 포함
    이 스테이지에서는 이미지의 전체적인 맥락과 배경에 대한 이해가 중요하며, 이는 특히 복잡한 의료 영상 분석에 있어서 매우 중요

7

U-Net

- U-Net
  확장 경로
  - 확장 경로는 U-Net의 오른쪽 부분을 형성하며, 수축 경로에서 얻은 특징 맵을 다시 원래 사이즈로 복원하는 역할

▼ 그림 5-42 확장 경로 세부 정보



8

U-Net

- U-Net
  - **업샘플링 층**: 확장 경로의 핵심 요소 중 하나는 업샘플링 층
    이 층들은 이미지의 차원을 점차적으로 확대하여, 수축 경로에서의 다운샘플링 과정으로 인해 손실된 상세 정보를 복구
    업샘플링은 보통 트랜스포즈 합성곱(transpose convolution)이나 최근접 이웃 업샘플링(nearest neighbor upsampling) 방법을 사용하여 수행
    이 과정에서 이미지의 사이즈는 증가하지만, 더 많은 픽셀이 생성되어 이미지의 상세 정보를 더욱 정밀하게 표현할 수 있게 됨
  - **합성곱 층**: 업샘플링된 특징 맵은 이후 합성곱 층을 거치게 됨
    이 층들은 업샘플링으로 확대된 이미지에서 부드러운 특징 맵을 생성하고, 이미지의 세부적인 부분을 더욱 세밀하게 다듬는 역할을 함
    합성곱 층은 이미지의 각 픽셀에 대한 정확한 분류를 가능하게 하며, 이는 최종적인 이미지 분할의 정밀도를 결정짓는 중요한 단계

9

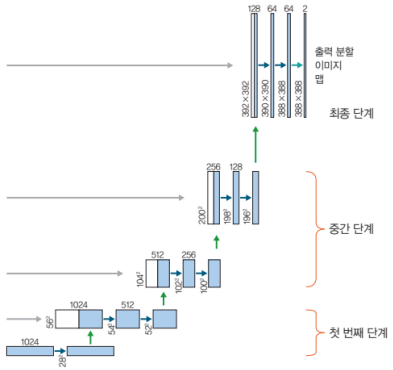
U-Net

- U-Net
  - **스킵 연결**(skip connections): 확장 경로에서는 스킵 연결이 중요한 기능
    이 연결은 수축 경로의 각 스테이지에서 얻은 특징 맵을 확장 경로의 해당 스테이지와 결합
    이러한 결합은 확장 경로에서 생성된 특징 맵에 수축 경로에서 추출된 상세한 위치 정보를 추가함으로써, 모델이 이미지의 정밀한 구조를 더 잘 이해하고 재현할 수 있게 함
    스킵 연결은 이미지의 깊은 특징과 표면적인 특징을 동시에 고려하게 하여, 더 정확하고 세밀한 이미지 분할 결과를 얻을 수 있게 해줌
    또한 확장 경로는 기본적으로 수축 경로의 역과정이라고 볼 수 있으며, 여기서는 업샘플링된 특징 맵이 수축 경로의 대응되는 특징 맵과 결합되어 세밀한 정보를 복원
    결합되는 단계의 기준은 수축 경로의 세부 단계 기준과 동일

10

U-Net

▼ 그림 5-43 확장 경로 세부 단계



11

U-Net

- U-Net
  - **첫 번째 단계**: 확장 경로의 첫 단계에서는 수축 경로의 마지막 합성곱 층의 출력을 받아 업샘플링을 시작
    여기서의 업샘플링은 주로 트랜스포즈 합성곱 연산을 통해 수행
    이렇게 업샘플링된 특징 맵은 수축 경로의 마지막 합성곱 층의 출력과 스킵 연결을 통해 결합
    이 결합은 이미지의 상세한 정보를 복원하는 데 중요한 역할을 하며, 업샘플링된 특징 맵에 수축 경로에서 추출된 위치 정보와 맥락 정보를 추가

12

U-Net

- U-Net
  - **중간 단계**: 확장 경로의 각 중간 스테이지에서는 계속해서 업샘플링과 스킵 연결을 통한 결합이 이루어짐
    각 스테이지에서는 업샘플링된 특징 맵을 수축 경로의 대응되는 스테이지의 출력과 결합
    이 과정은 네트워크가 수축 경로에서 잃어버린 상세한 정보를 되찾고, 동시에 이미지의 맥락을 유지할 수 있게 함
    중간 스테이지의 각 단계는 이미지의 다양한 수준의 특징을 세밀하게 재구성하며, 이는 정밀한 이미지 분할에 결정적임
  - **마지막 단계**: 확장 경로의 마지막 단계에서는 최종 업샘플링과 결합이 이루어짐
    이 단계는 이미지를 원래의 해상도로 복원하며, 수축 경로의 첫 번째 스테이지에서 얻은 특징 맵과 결합
    마지막 스테이지의 결합은 이미지의 가장 상세한 정보를 복원하는 데 중요하며, 이를 통해 네트워크는 이미지의 세밀한 경계와 구조를 정확하게 재현할 수 있음

13

U-Net

- U-Net

텐서플로를 활용한 U-Net 실습

- U-Net의 핵심 구성 요소인 수축 경로와 확장 경로의 작동 원리를 실습하면서 직접 경험하고 이해해보자
- 수축 경로에서는 이미지의 중요한 특징을 추출하는 방법을, 확장 경로에서는 이러한 특징들을 사용하여 이미지를 원래 사이즈로 복원하는 과정을 배우게 됨
- 특히 스킵 연결을 통해 수축 경로와 확장 경로 사이의 정보를 효율적으로 결합하는 방법을 실습하며, 이는 U-Net이 세밀한 이미지 분할을 수행하는 데 결정적인 역할을 하는 부분

```
!pip install git+https://github.com/tensorflow/examples.git

import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
from tensorflow_examples.models.pix2pix import pix2pix
from IPython.display import clear_output
```

14

U-Net

- U-Net
  - 다음으로 훈련 및 검증 데이터 세트를 준비하는 과정을 설정

```
dataset, info = tfds.load('oxford_iiit_pet:3.*.*', with_info=True) # ①

def normalize(input_image, input_mask): # ②
    input_image = tf.cast(input_image, tf.float32) / 255.0
    input_mask -= 1
    return input_image, input_mask

def load_image(datapoint): # ③
    input_image = tf.image.resize(datapoint['image'], (128, 128))
    input_mask = tf.image.resize(
        datapoint['segmentation_mask'],
        (128, 128),
        method = tf.image.ResizeMethod.NEAREST_NEIGHBOR,
  )
    input_image, input_mask = normalize(input_image, input_mask)

    return input_image, input_mask
```

15

U-Net

- U-Net
  - 다음처럼 훈련 및 테스트 이미지를 로딩하는 과정을 설정

```
TRAIN_LENGTH = info.splits['train'].num_examples
BATCH_SIZE = 64
BUFFER_SIZE = 1000
STEPS_PER_EPOCH = TRAIN_LENGTH // BATCH_SIZE

train_images = dataset['train'].map(load_image, num_parallel_calls=tf.data.AUTOTUNE)
test_images = dataset['test'].map(load_image, num_parallel_calls=tf.data.AUTOTUNE)
```

16

U-Net

- U-Net
  - 다음은 데이터 증강 기법을 적용하여 훈련 데이터의 다양성을 높이겠음

```
class Augment(tf.keras.layers.Layer): # ①
    def __init__(self, seed=42):
        super().__init__()
        self.augment_inputs = tf.keras.layers.RandomFlip(mode="horizontal", seed=seed)
        self.augment_labels = tf.keras.layers.RandomFlip(mode="horizontal", seed=seed)

    def call(self, inputs, labels):
        inputs = self.augment_inputs(inputs)
        labels = self.augment_labels(labels)
        return inputs, labels
```

17

U-Net

- U-Net

```
train_batches = (
    train_images
    .cache()
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE)
    .repeat()
    .map(Augment())
    .prefetch(buffer_size=tf.data.AUTOTUNE)) # ②

test_batches = test_images.batch(BATCH_SIZE)
```

18

U-Net

- U-Net

① **Augment 클래스 정의**

__init__ 메서드에서는 RandomFlip 층을 사용하여 이미지를 수평으로 무작위로 뒤집는 증강을 정의

동일한 시드(seed)를 사용하여 입력 이미지와 레이블 이미지에 동일한 변형을 적용

call 메서드는 실제로 데이터 증강을 수행하는 함수로, 입력된 이미지와 레이블에 RandomFlip을 적용

19

U-Net

- U-Net

② **훈련 데이터 배치 설정**

- train_batches: 훈련 데이터 세트를 처리하여 모델 학습에 적합한 형태로 만듦
- .cache(): 데이터 세트를 캐시에 저장하여, 각 에포크에서 데이터를 다시 로드하는 시간을 줄임
- .shuffle(BUFFER_SIZE): 데이터 세트를 섞어, 모델이 특정 순서에 의존하지 않도록 함
- .batch(BATCH_SIZE): 데이터 세트를 지정된 배치 사이즈로 분할
- .repeat(): 데이터 세트를 여러 번 반복하여 여러 에포크 동안 사용할 수 있게 함
- .map(Augment()): 앞서 정의한 Augment 클래스를 적용하여 데이터 증강을 수행
- .prefetch(buffer_size=tf.data.AUTOTUNE): 학습 중 데이터 로딩 시간을 줄이기 위해 데이터를 미리 가져옴

  tf.data.AUTOTUNE은 텐서플로가 자동으로 런타임에 최적의 버퍼 사이즈를 결정

20

U-Net

- U-Net
  - 이제 학습 데이터를 시각화하여 마스크 데이터 세트와 예측 데이터 세트에 대한 설정을 진행해보자

```
def display(display_list):
    plt.figure(figsize=(15, 15))

    title = ['Input Image', 'True Mask', 'Predicted Mask']

    for i in range(len(display_list)):
        plt.subplot(1, len(display_list), i+1)
        plt.title(title[i])
        plt.imshow(tf.keras.utils.array_to_img(display_list[i]))
        plt.axis('off')
    plt.show()

for images, masks in train_batches.take(2):
    sample_image, sample_mask = images[0], masks[0]
    display([sample_image, sample_mask])
```

21

U-Net

- U-Net

22

U-Net

- U-Net
  - 이번 백본 네트워크는 모바일넷 v2를 사용하여 진행

```
base_model = tf.keras.applications.MobileNetV2(input_shape=[128, 128, 3], include_
top=False) # ①

layer_names = [
    'block_1_expand_relu',   # 64x64
    'block_3_expand_relu',   # 32x32
    'block_6_expand_relu',   # 16x16
    'block_13_expand_relu',  # 8x8
    'block_16_project',      # 4x4
] # ②
base_model_outputs = [base_model.get_layer(name).output for name in layer_names] # ③

down_stack = tf.keras.Model(inputs=base_model.input, outputs=base_model_outputs) # ④

down_stack.trainable = False
```

23

U-Net

- U-Net
  - 이어서 확장 경로, U-Net 모델을 완성해보자

```
up_stack = [
    pix2pix.upsample(512, 3),  # 4x4 -> 8x8
    pix2pix.upsample(256, 3),  # 8x8 -> 16x16
    pix2pix.upsample(128, 3),  # 16x16 -> 32x32
    pix2pix.upsample(64, 3),   # 32x32 -> 64x64
] # ①

def U-NET_model(output_channels:int): # ②
    inputs = tf.keras.layers.Input(shape=[128, 128, 3])

    skips = down_stack(inputs)
    x = skips[-1] # ③
    skips = reversed(skips[:-1])
```

24

U-Net

- U-Net

```
    for up, skip in zip(up_stack, skips):
        x = up(x)
        concat = tf.keras.layers.Concatenate()
        x = concat([x, skip])

    last = tf.keras.layers.Conv2DTranspose(
        filters=output_channels, kernel_size=3, strides=2, padding='same') # ④
        # 64x64 -> 128x128
    x = last(x)

return tf.keras.Model(inputs=inputs, outputs=x)
```

25

U-Net

- U-Net
  - 이제 컴파일을 진행한 후 모델을 시각화해보자

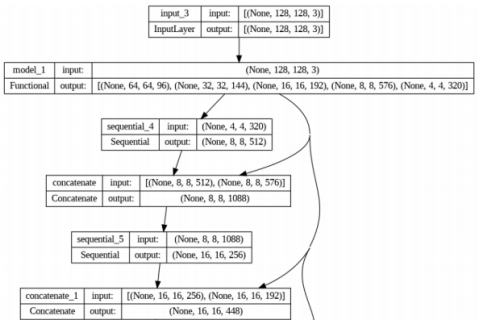
```
OUTPUT_CLASSES = 3

model = U-NET_model(output_channels=OUTPUT_CLASSES)
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
tf.keras.utils.plot_model(model, show_shapes=True)
```

26

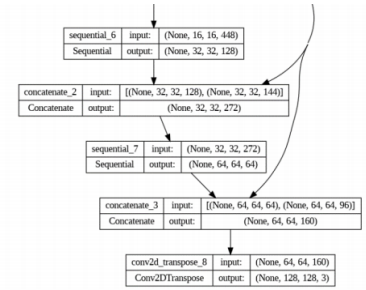
U-Net

▼ 그림 5-44 출력 결과: U-Net Summary



27

U-Net



28

U-Net

- U-Net
  - 이제 모델이 학습하며 모델의 예측 결과를 시각화하는 함수를 정의하고, 모델을 학습시키는 과정을 진행해보자

```
def create_mask(pred_mask): # ①
    pred_mask = tf.math.argmax(pred_mask, axis=-1)
    pred_mask = pred_mask[..., tf.newaxis]
    return pred_mask[0]

def show_predictions(dataset=None, num=1): # ②
    if dataset:
        for image, mask in dataset.take(num):
            pred_mask = model.predict(image)
            display([image[0], mask[0], create_mask(pred_mask)])
    else:
        display([sample_image, sample_mask,
            create_mask(model.predict(sample_image[tf.newaxis, ...]))])
```

29

U-Net

- U-Net

```
class DisplayCallback(tf.keras.callbacks.Callback): # ③
    def on_epoch_end(self, epoch, logs=None):
        clear_output(wait=True)
        show_predictions()
        print ('\nSample Prediction after epoch {}\n'.format(epoch+1))

EPOCHS = 20
VAL_SUBSPLITS = 5
VALIDATION_STEPS = info.splits['test'].num_examples//BATCH_SIZE//VAL_SUBSPLITS

model_history = model.fit(train_batches, epochs=EPOCHS,
                          steps_per_epoch=STEPS_PER_EPOCH,
                          validation_steps=VALIDATION_STEPS,
                          validation_data=test_batches,
                          callbacks=[DisplayCallback()]) # ④
```

30

U-Net

▼ 그림 5-45 출력 결과: U-Net 학습 결과와 실제 정답 마스크 비교

Sample Prediction after epoch 20

31