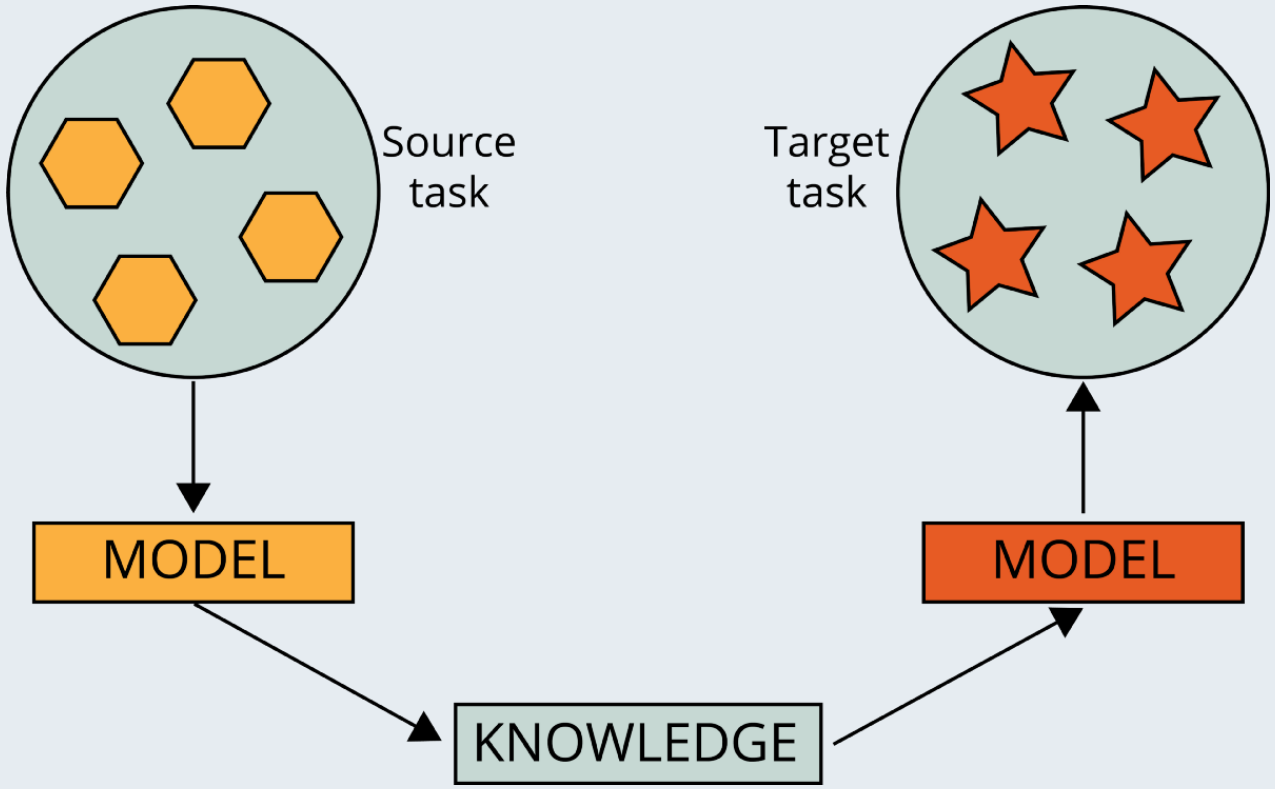
Transfer Learning(전이학습)



박경규

Data Augmentation(데이터 증강)

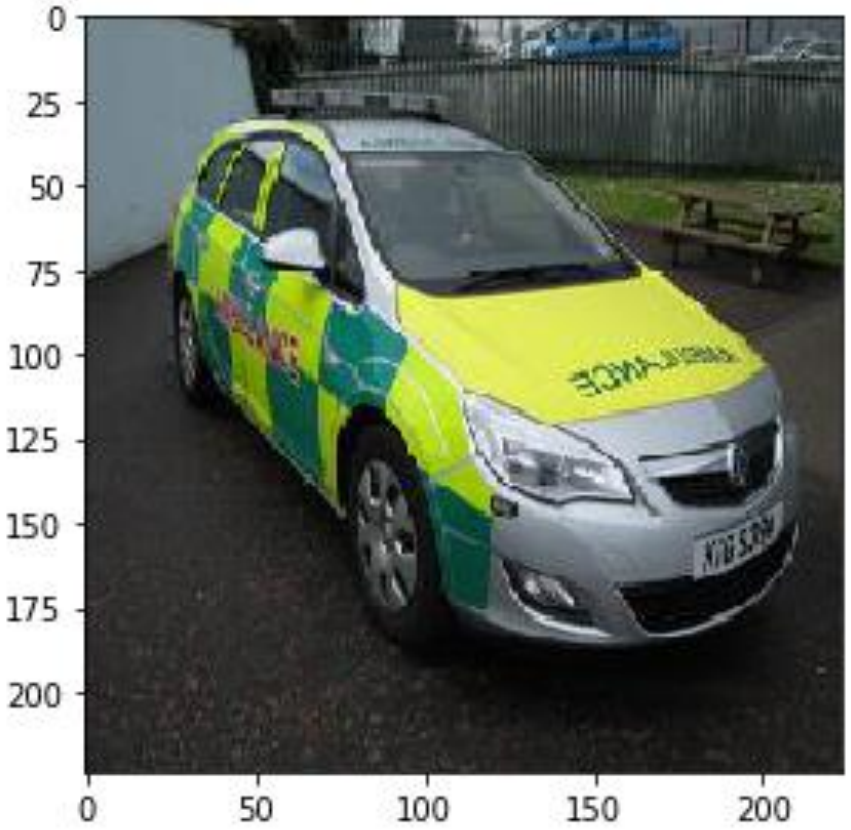
- Image Augmentation은 딥 러닝 모델을 훈련하기 위해 새로운 이미지를 생성하는 프로세스입니다.
- 이러한 새 이미지는 기존 학습 이미지를 사용하여 생성되므로 수동으로 수집 할 필요가 없습니다.

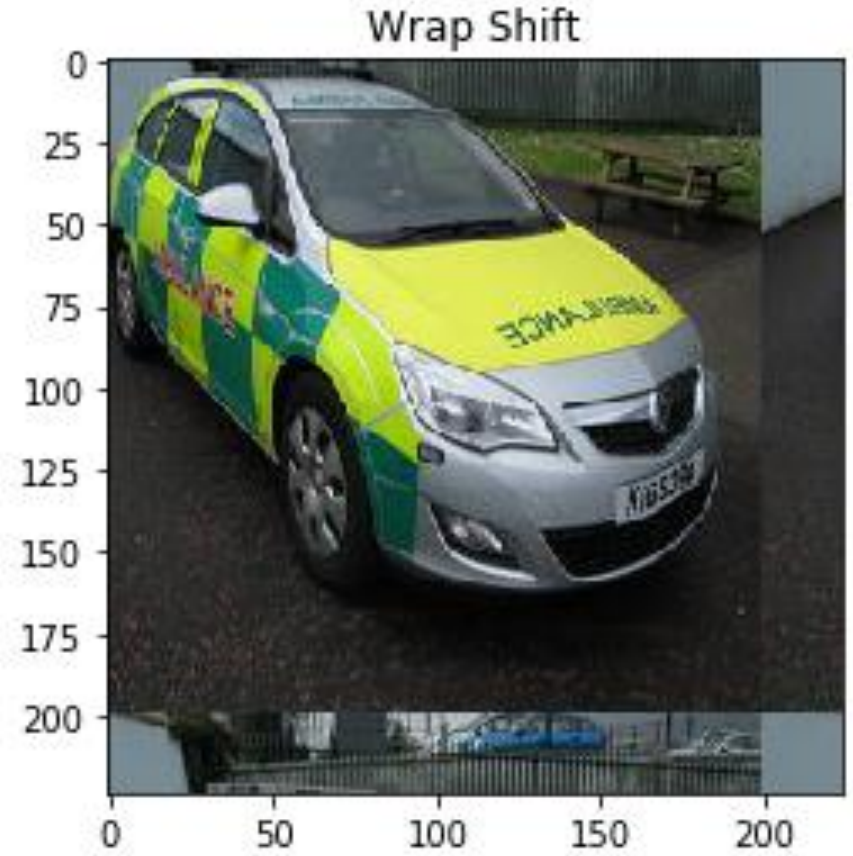
https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Data Augmentation – Rotation

https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

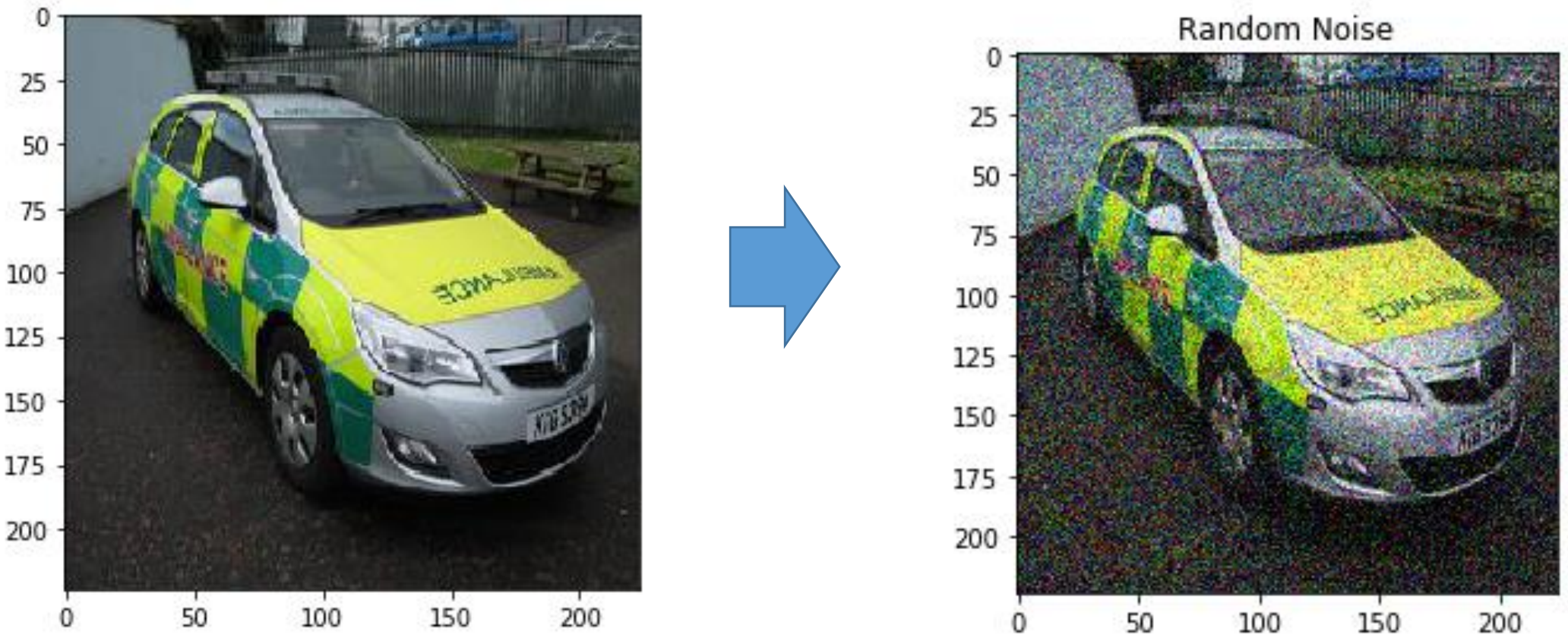
Data Augmentation – Shifting





https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Data Augmentation – Adding Noise

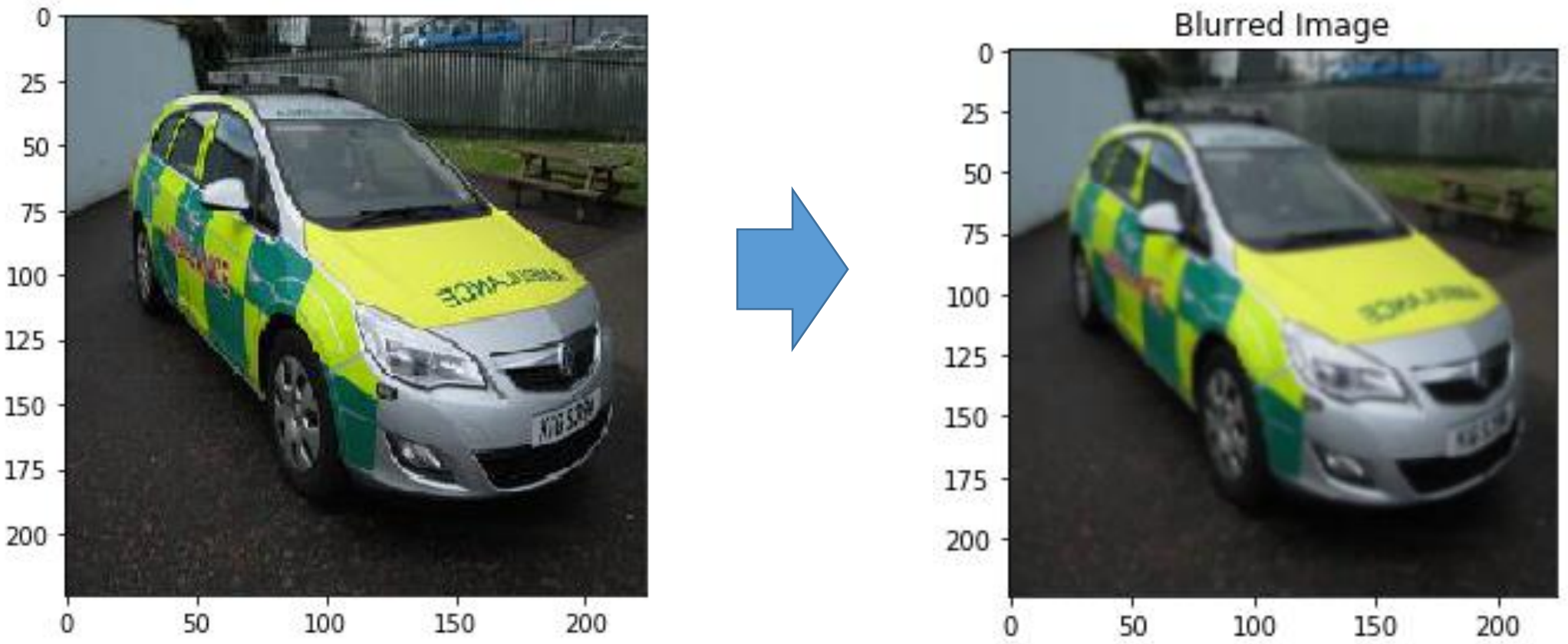


https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Data Augmentation – Shifting

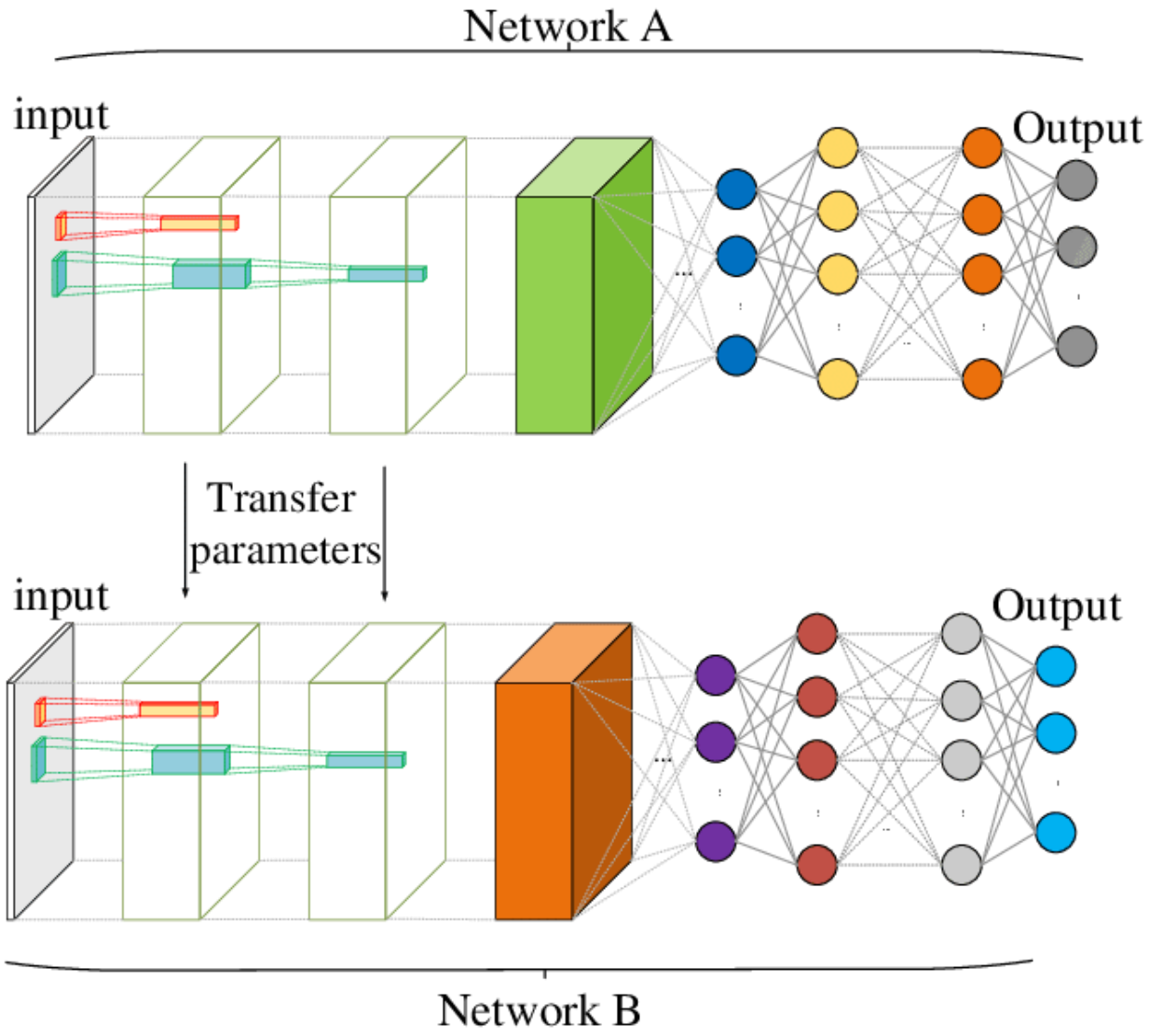
https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

Data Augmentation – Blurring

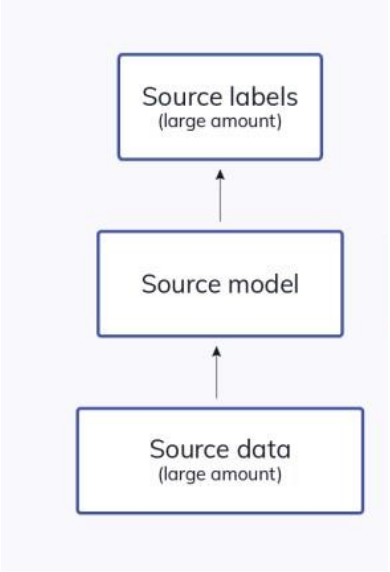


https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/

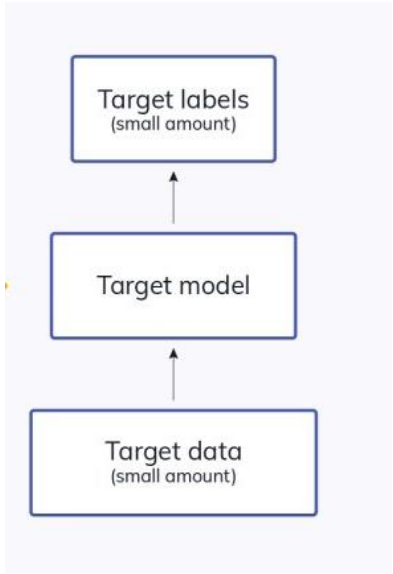
Transfer Learning 개요



https://www.researchgate.net/figure/Illustration-of-transfer-learning-concept-where-the-rst-layers-in-network-A-and-network-B_fig2_316748306



Transfer Learned Knowledge



Transfer Learning 시나리오

■ 전이학습

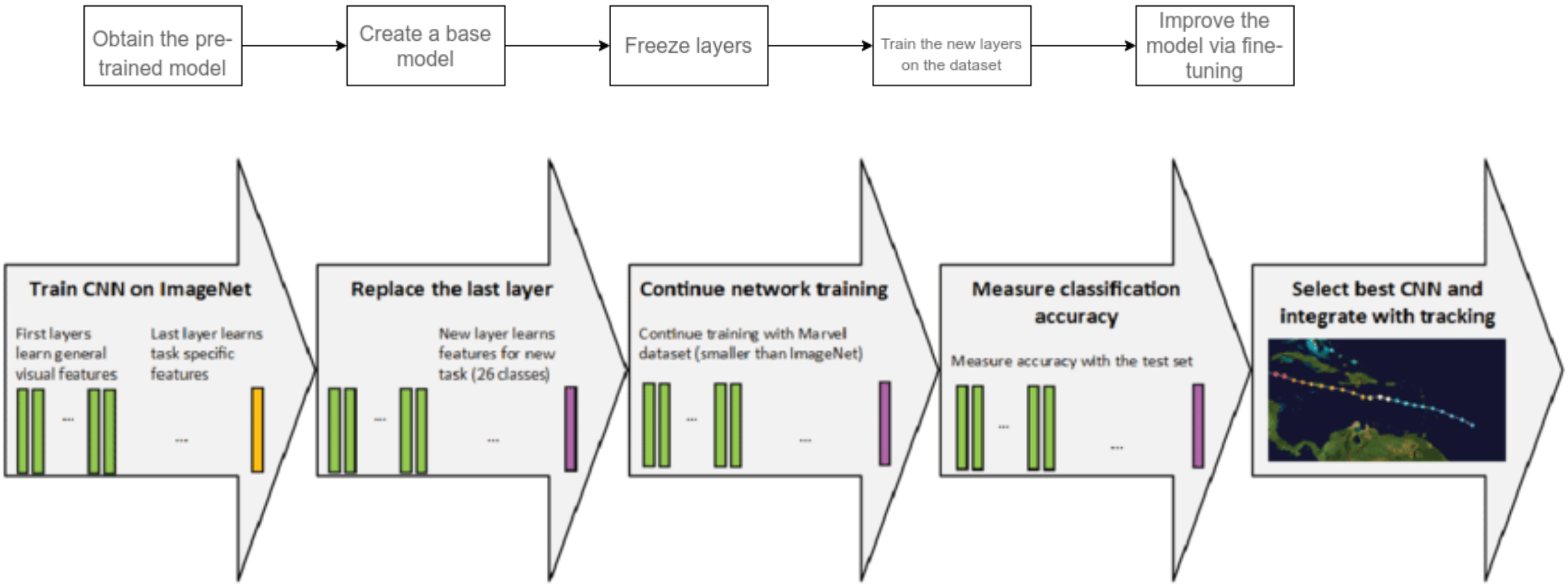
- 실제로 충분한 크기의 데이터셋을 갖추기는 상대적으로 드물기 때문에,
  (무작위 초기화를 통해) 맨 처음부터 합성곱 신경망(Convolutional Network) 전체를
  학습하는 사람은 매우 적습니다.
- 대신, 매우 큰 데이터셋(예. 100가지 분류에 대해 120만개의 이미지가 포함된 ImageNet)에서
  합성곱 신경망(ConvNet)을 미리 학습한 후, 이 합성곱 신경망을 관심있는 작업을 위한 초기 설정
  또는 고정된 특징 추출기(fixed feature extractor)로 사용합니다.

■ 전이학습 시나리오

- 고정된 특징 추출기로써의 합성곱 신경망: 여기서는 마지막에 완전히 연결 된 계층을 제외한 모든 신경망의
  가중치를 고정합니다.
  마지막의 FCN(Fully Connected Network)은 새로운 무작위의 가중치를 갖는 계층으로 대체되어
  이 계층만 학습합니다.
- 합성곱 신경망의 미세조정(Fine Tuning): 무작위 초기화 대신, 신경망을 ImageNet 1000 데이터셋 등으로
  미리 학습한 신경망으로 초기화합니다. 학습의 나머지 과정들은 평상시와 같습니다.

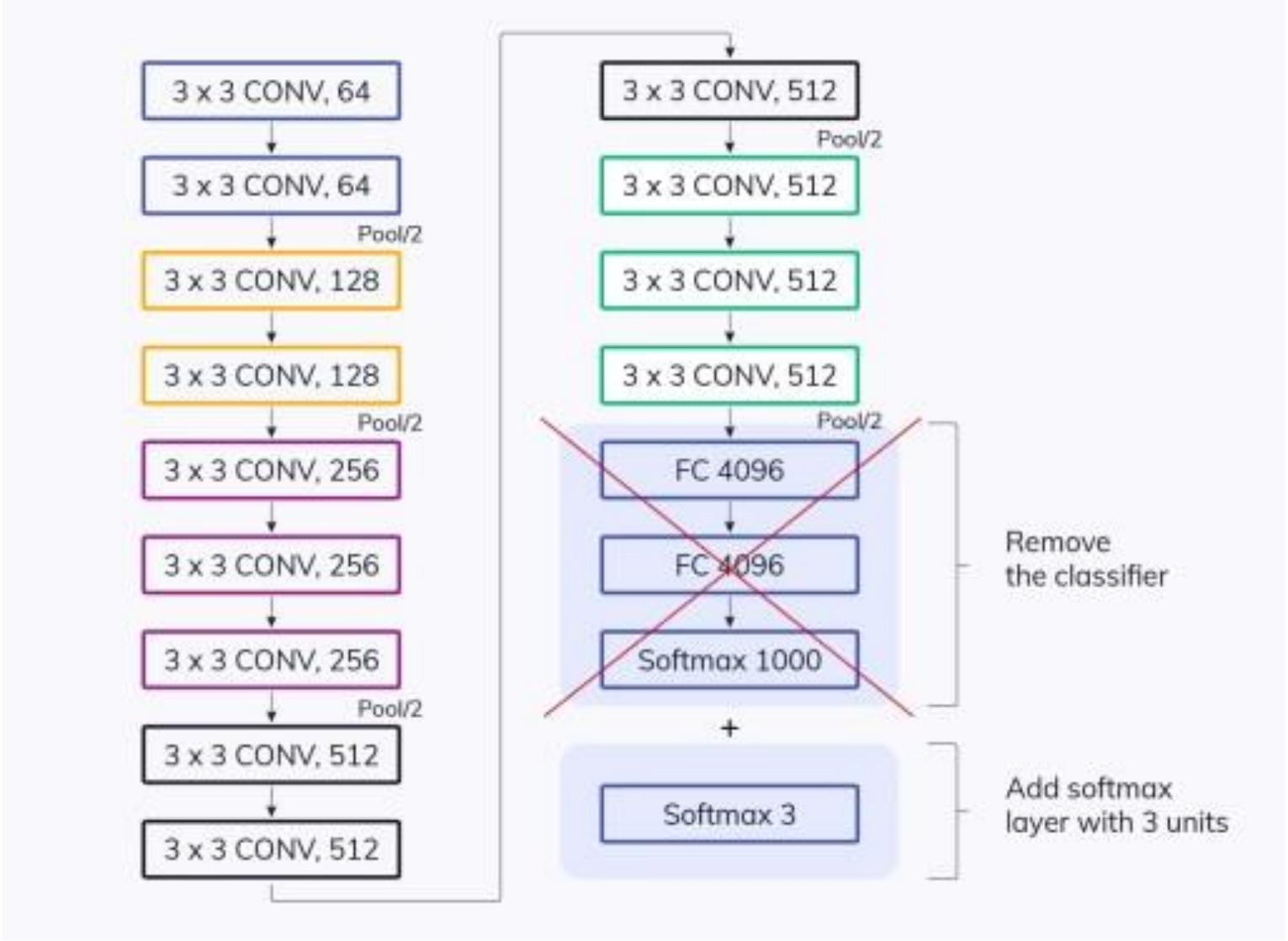
https://tutorials.pytorch.kr/beginner/transfer_learning_tutorial.html

Transfer Learning 절차



https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

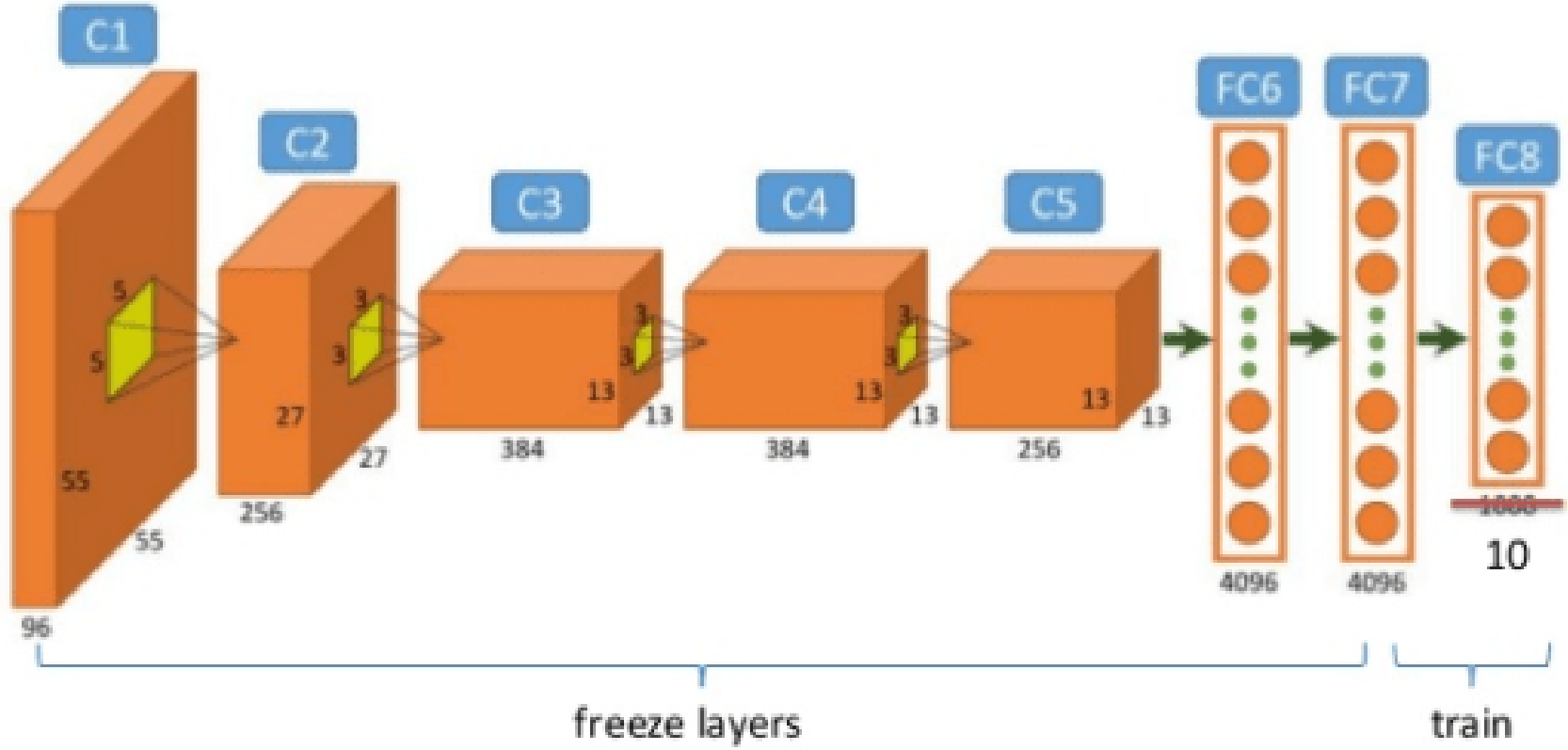
Transfer Learning 절차 – Create a base model



https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

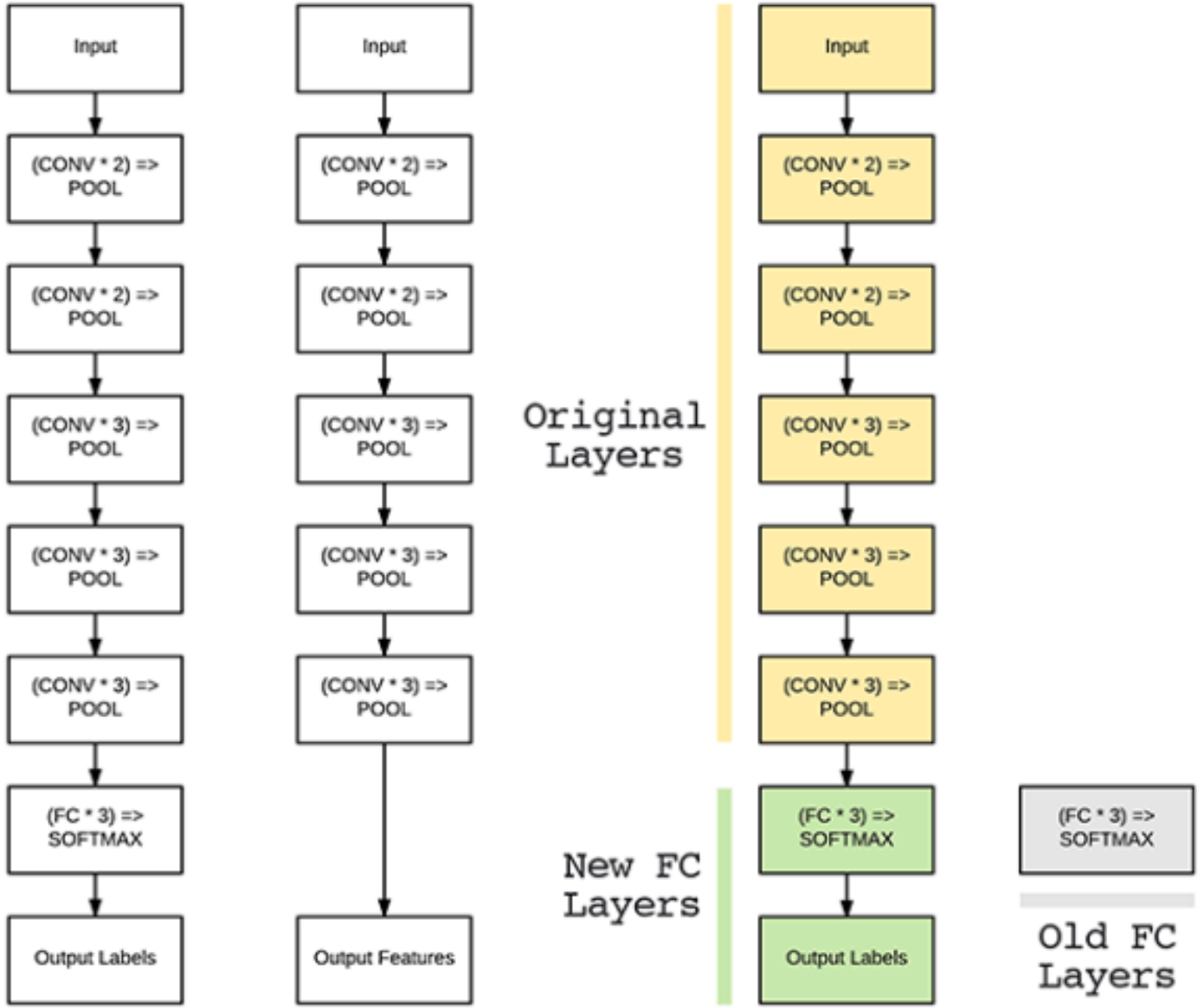
Transfer Learning 절차 – Freeze layers

```
base_model.trainable = False
```



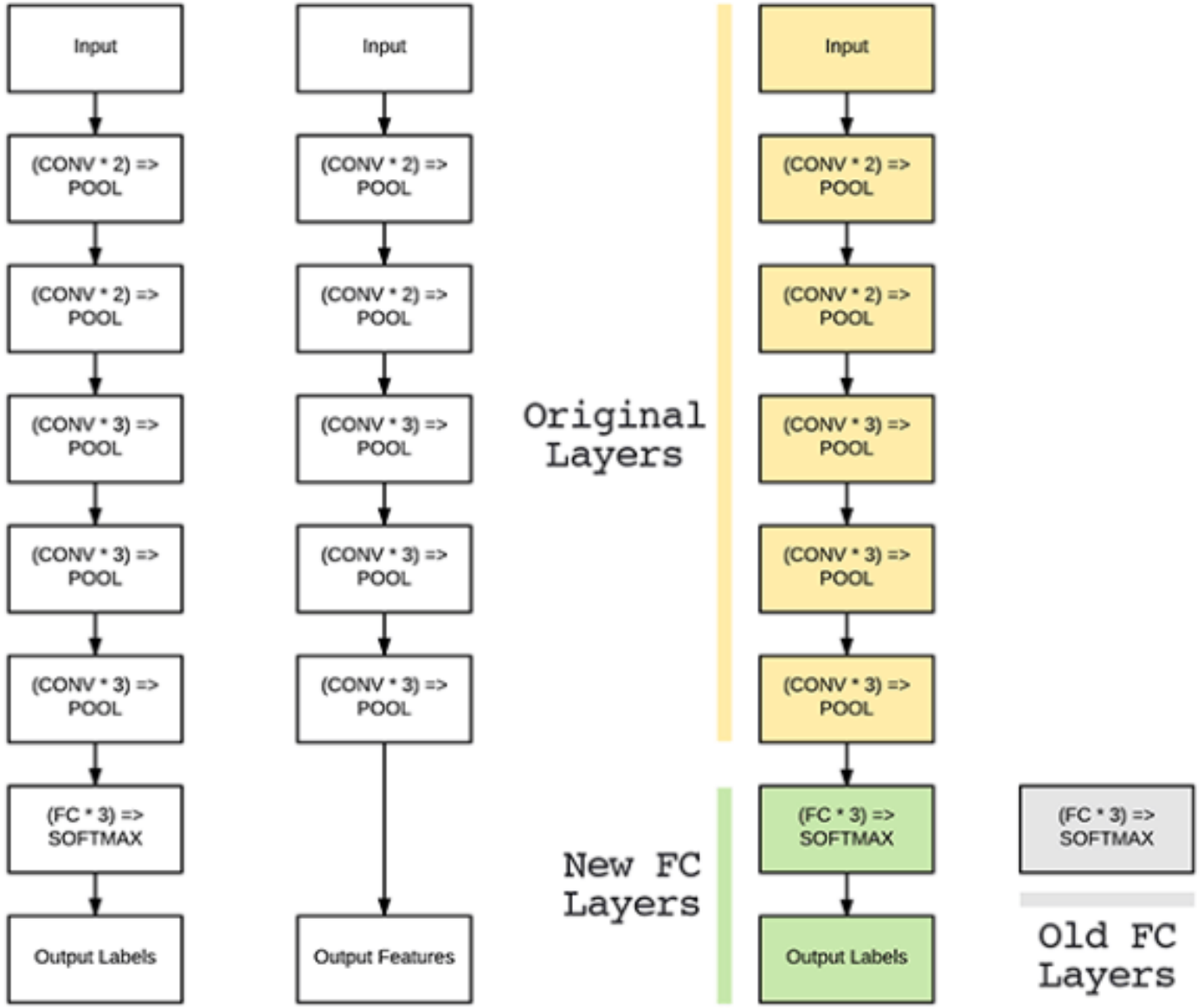
https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

Transfer Learning 절차 – Add new trainable layers



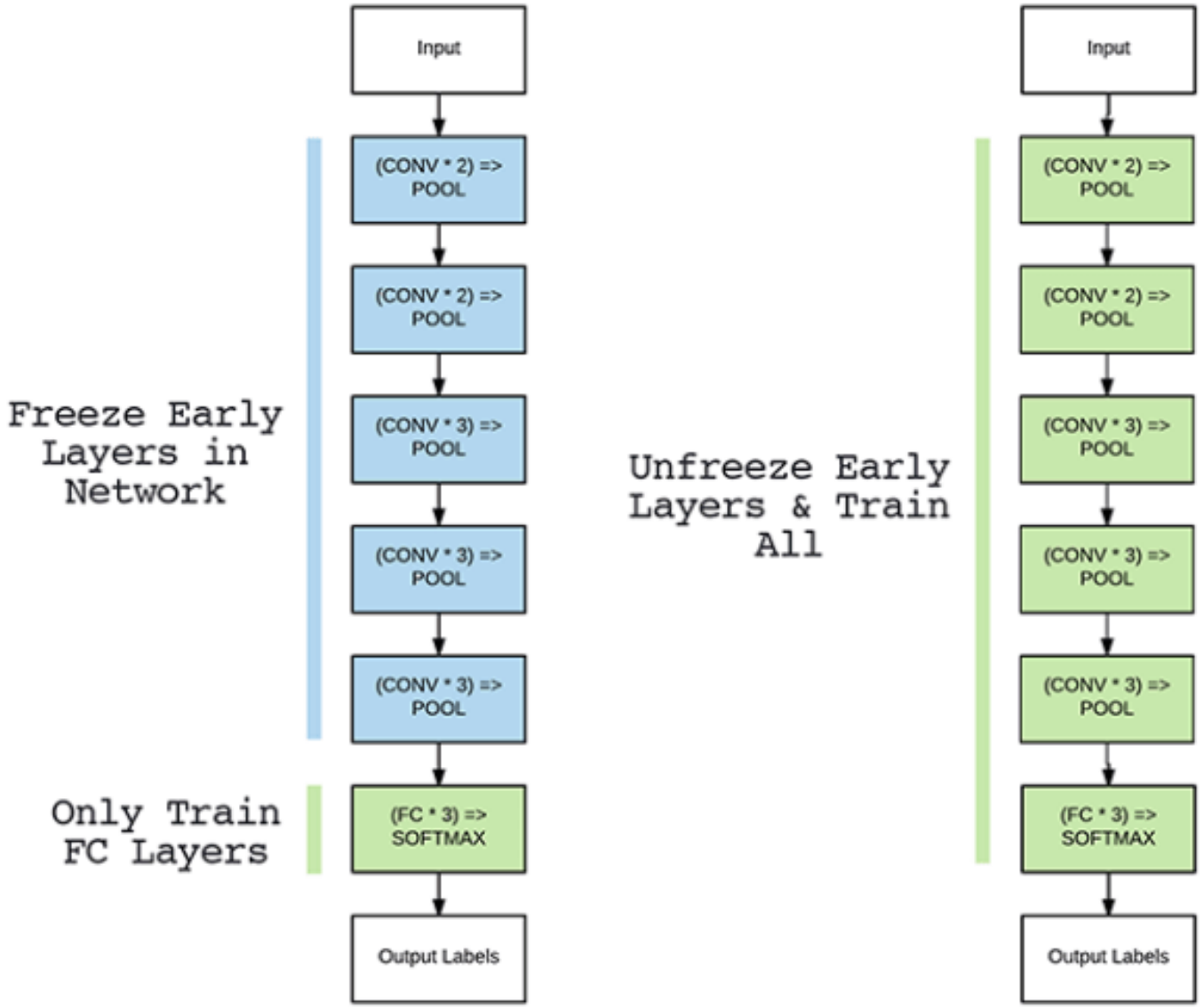
https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

Transfer Learning 절차 – Train the new layers on the dataset



https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

Transfer Learning 절차 – Improve the model via fine-tuning



https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

Pre-trained Model : tf.keras.applications

https://www.tensorflow.org/api_docs/python/tf/keras/applications

`inception_resnet_v2` module: Public API for tf.keras.applications.inception_resnet_v2 namespace.

`inception_v3` module: Public API for tf.keras.applications.inception_v3 namespace.

`mobilenet` module: Public API for tf.keras.applications.mobilenet namespace.

`mobilenet_v2` module: Public API for tf.keras.applications.mobilenet_v2 namespace.

`mobilenet_v3` module: Public API for tf.keras.applications.mobilenet_v3 namespace.

`nasnet` module: Public API for tf.keras.applicatio

`resnet` module: Public API for tf.keras.applicatio

`resnet50` module: Public API for tf.keras.applicat

`resnet_v2` module: Public API for tf.keras.applic

`vgg16` module: Public API for tf.keras.application

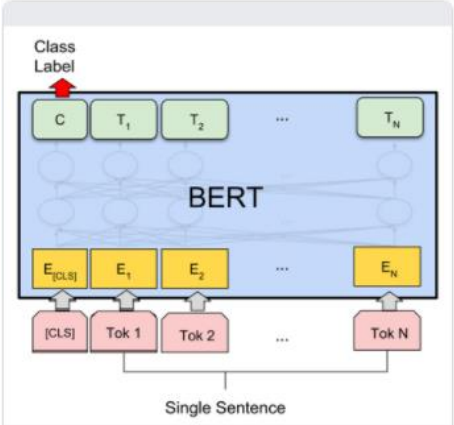
`vgg19` module: Public API for tf.keras.application

`xception` module: Public API for tf.keras.applicat

```
model = tf.keras.applications.MobileNet(
    input_shape=None,
    alpha=1.0,
    depth_multiplier=1,
    dropout=0.001,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

Pre-trained Model : TensorFlow Hub

https://www.tensorflow.org/hub



BERT

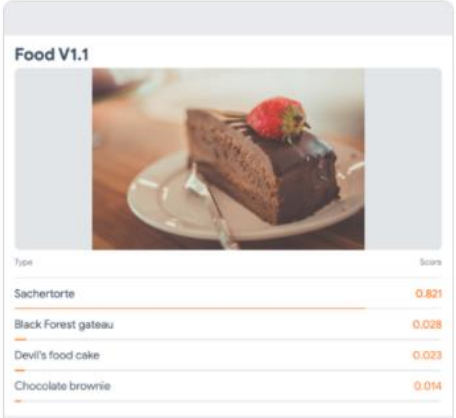
텍스트 분류 및 질문 답변 등 NLP 작업에서 BERT를 확인해 보세요.

객체 감지

이미지에서 객체를 감지하려면 Faster R-CNN Inception ResNet V2 640x640 모델을 사용하세요.

스타일 전이

이미지 하나의 스타일을 이미지 스타일 전이 모델을 사용하여 다른 이미지로 전이하세요.



기기 내 음식 분류 기준

이 TFLite 모델을 사용하여 휴대기기의 음식 사진을 분류하세요.

```
model = tf.keras.Sequential([
 hub.KerasLayer("https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4",
                trainable=False),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

Transfer learning and fine-tuning

https://bit.ly/3p6BuR2

- 160x160픽셀의 수천 개의 고양이와 개의 이미지가 포함된 데이터세트를 사용합니다.

Transfer learning and fine-tuning

```
[4]  _URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'
     path_to_zip = tf.keras.utils.get_file('cats_and_dogs.zip', origin=_URL, extract=True)
     PATH = os.path.join(os.path.dirname(path_to_zip), 'cats_and_dogs_filtered')

     train_dir = os.path.join(PATH, 'train')
     validation_dir = os.path.join(PATH, 'validation')

     BATCH_SIZE = 32
     IMG_SIZE = (160, 160)

     train_dataset = image_dataset_from_directory(train_dir,
                                                  shuffle=True,
                                                  batch_size=BATCH_SIZE,
                                                  image_size=IMG_SIZE)
```

```
Downloading data from https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip
68608000/68606236 [==============================] - 1s 0us/step
68616192/68606236 [==============================] - 1s 0us/step
Found 2000 files belonging to 2 classes.
```

Transfer learning and fine-tuning

원본 데이터세트에는 테스트 세트가 포함되어 있지 않으므로 테스트 세트를 생성합니다.
`tf.data.experimental.cardinality`를 사용하여 검증 세트에서 사용할 수 있는
데이터 배치 수를 확인한 다음 그 중 20%를 테스트 세트로 이동합니다.

```
[7]  val_batches = tf.data.experimental.cardinality(validation_dataset)
     test_dataset = validation_dataset.take(val_batches // 5)
     validation_dataset = validation_dataset.skip(val_batches // 5)
```

```
[8]  print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation_dataset))
     print('Number of test batches: %d' % tf.data.experimental.cardinality(test_dataset))
```

```
Number of validation batches: 26
Number of test batches: 6
```

Transfer learning and fine-tuning

성능을 높이도록 데이터세트 구성하기

버퍼링된 프리페치를 사용하여 I/O 차단 없이 디스크에서 이미지를 로드합니다.
이 방법에 대해 자세히 알아보려면 데이터 성능 가이드를 참조하세요.

```
[9]  AUTOTUNE = tf.data.AUTOTUNE

     train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
     validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
     test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

Transfer learning and fine-tuning

데이터 증강 사용

큰 이미지 데이터세트가 없는 경우, 회전 및 수평 뒤집기와 같이 훈련 이미지에 무작위이지만 사실적인 변환을 적용하여 샘플 다양성을 인위적으로 도입하는 것이 좋습니다.
이것은 모델을 훈련 데이터의 다양한 측면에 노출시키고 과대적합을 줄이는 데 도움이 됩니다.
이 튜토리얼에서 데이터 증강에 대해 자세히 알아볼 수 있습니다.

```
[10]  data_augmentation = tf.keras.Sequential([
        tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
        tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
      ])
```

참고: `model.fit`을 호출할 때 훈련 중에만 이러한 레이어가 활성화됩니다.
`model.evaulate` 또는 `model.fit`의 추론 모드에서 모델을 사용하면 비활성화됩니다.

Transfer learning and fine-tuning

사전 훈련된 컨볼루션 네트워크로부터 기본 모델 생성하기

Google에서 개발한 MobileNet V2 모델로부터 기본 모델을 생성합니다.
이 모델은 1.4M 이미지와 1000개의 클래스로 구성된 대규모 데이터셋인
ImageNet 데이터셋를 사용해 사전 훈련된 모델입니다.

```
[14] # Create the base model from the pre-trained model MobileNet V2
     IMG_SHAPE = IMG_SIZE + (3,)
     base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                                    include_top=False,
                                                    weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobi
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
```

Transfer learning and fine-tuning

이 특징 추출기는 각 160x160x3 이미지를 5x5x1280 개의 특징 블록으로 변환합니다.
이미지 배치 예제에서 수행하는 작업을 확인하세요:

```
[15]  image_batch, label_batch = next(iter(train_dataset))
      feature_batch = base_model(image_batch)
      print(feature_batch.shape)
```

```
(32, 5, 5, 1280)
```

컨볼루션 베이스 모델 고정하기

모델을 컴파일하고 훈련하기 전에 컨볼루션 기반을 고정하는 것이 중요합니다.

```
[16]  base_model.trainable = False
```

Transfer learning and fine-tuning

분류 층을 맨 위에 추가하기

특성 블록에서 예측을 생성하기 위해 `tf.keras.layers.GlobalAveragePooling2D` 레이어를 사용하여 특성을 이미지당 하나의 1280-요소 벡터로 변환하여 `5x5` 공간 위치에 대한 평균을 구합니다.

```
[18]  global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
      feature_batch_average = global_average_layer(feature_batch)
      print(feature_batch_average.shape)
```

```
(32, 1280)
```

Transfer learning and fine-tuning

`tf.keras.layers.Dense` 레이어를 사용하여 특성을 이미지당 단일 예측으로 변환합니다.

```
[56]  prediction_layer = tf.keras.layers.Dense(1)
      prediction_batch = prediction_layer(feature_batch_average)
      print(prediction_batch.shape)
```

```
(32, 1)
```

Transfer learning and fine-tuning

Keras Functional API를 사용하여 데이터 증강, 크기 조정, base_model 및 특성 추출기 레이어를 함께 연결하여 모델을 구축합니다.
앞서 언급했듯이 모델에 BatchNormalization 레이어가 포함되어 있으므로 training=False를 사용하세요.

```
[57]  inputs = tf.keras.Input(shape=(160, 160, 3))
      x = data_augmentation(inputs)
      x = preprocess_input(x)
      x = base_model(x, training=False)
      x = global_average_layer(x)
      x = tf.keras.layers.Dropout(0.2)(x)
      outputs = prediction_layer(x)
      model = tf.keras.Model(inputs, outputs)
```

Transfer learning and fine-tuning

모델 컴파일

학습하기 전에 모델을 컴파일해야 합니다.
두 개의 클래스가 있으므로 모델이 선형 출력을 제공하므로
`from_logits = True`와 함께 이진 교차 엔트로피 손실을 사용하세요.

```
[58]  base_learning_rate = 0.0001
      model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=base_learning_rate),
                    loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                    metrics=['accuracy'])
```

Transfer learning and fine-tuning

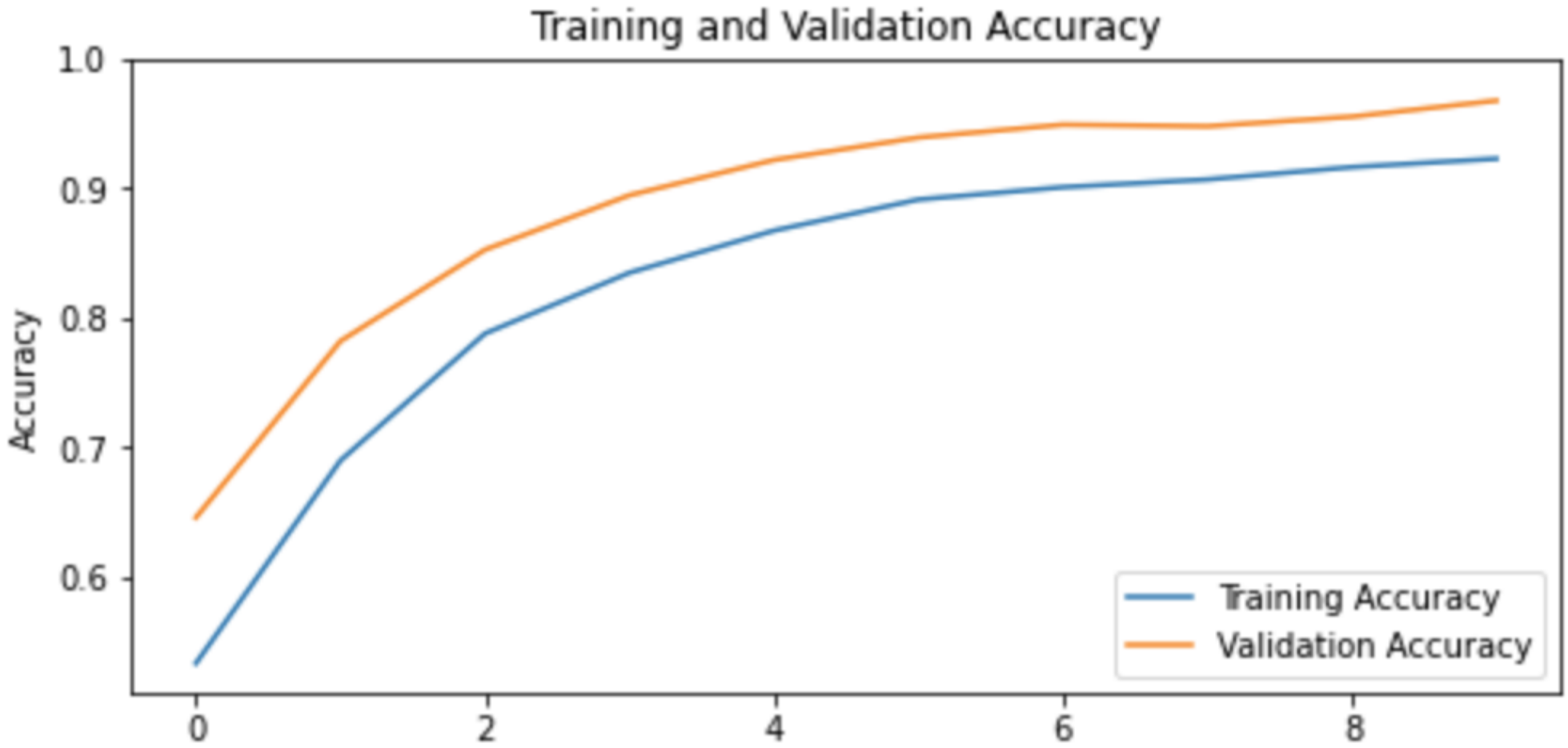
모델 훈련

10 epoch만큼 훈련한 후, 검증 세트에서 ~94%의 정확도를 볼 수 있습니다.

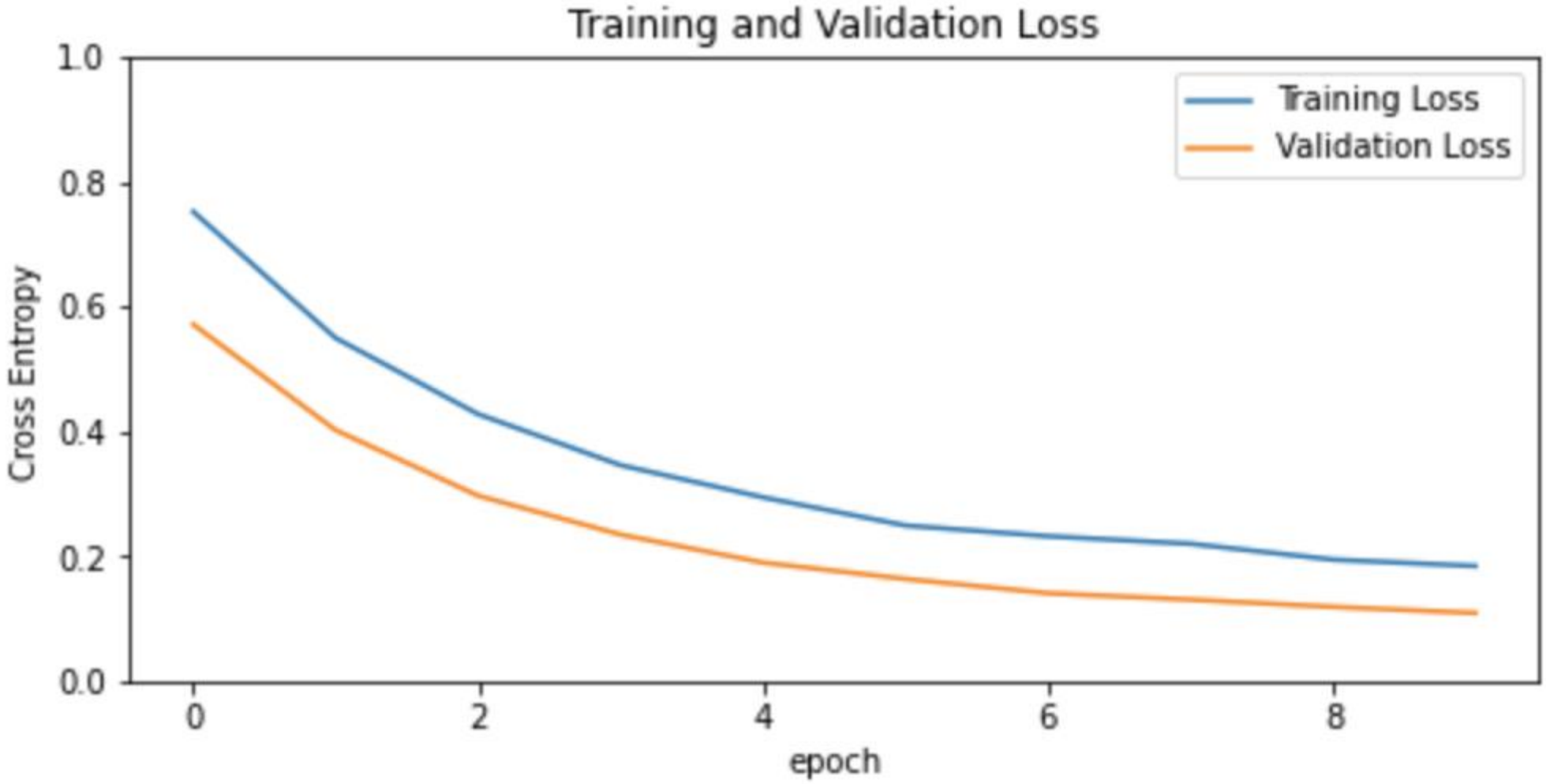
```
[63] history = model.fit(train_dataset,
                         epochs=initial_epochs,
                         validation_data=validation_dataset)
```

```
Epoch 6/10
63/63 [==============================] - 6s 86ms/step - loss: 0.2491 - accuracy: 0.8915 - val_loss: 0.1634 - val_accuracy: 0.9394
Epoch 7/10
63/63 [==============================] - 6s 88ms/step - loss: 0.2319 - accuracy: 0.9010 - val_loss: 0.1405 - val_accuracy: 0.9493
Epoch 8/10
63/63 [==============================] - 6s 87ms/step - loss: 0.2198 - accuracy: 0.9070 - val_loss: 0.1302 - val_accuracy: 0.9480
Epoch 9/10
63/63 [==============================] - 6s 87ms/step - loss: 0.1943 - accuracy: 0.9165 - val_loss: 0.1186 - val_accuracy: 0.9554
Epoch 10/10
63/63 [==============================] - 6s 87ms/step - loss: 0.1841 - accuracy: 0.9230 - val_loss: 0.1087 - val_accuracy: 0.9678
```

Transfer learning and fine-tuning



Transfer learning and fine-tuning



Transfer learning and fine-tuning

https://www.tensorflow.org/tutorials/images/transfer_learning?hl=ko

Transfer learning with TensorFlow Hub

https://www.tensorflow.org/tutorials/images/transfer_learning_with_hub?hl=ko

Transfer Learning Guide

https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

THANK YOU

kgpark88@gmail.com