Fine-tuning(Transfer Learning)

- 이미 학습된 Neural Network를 새로운 Task에 맞게 다시 미세조정(Fine-Tuning)한다.

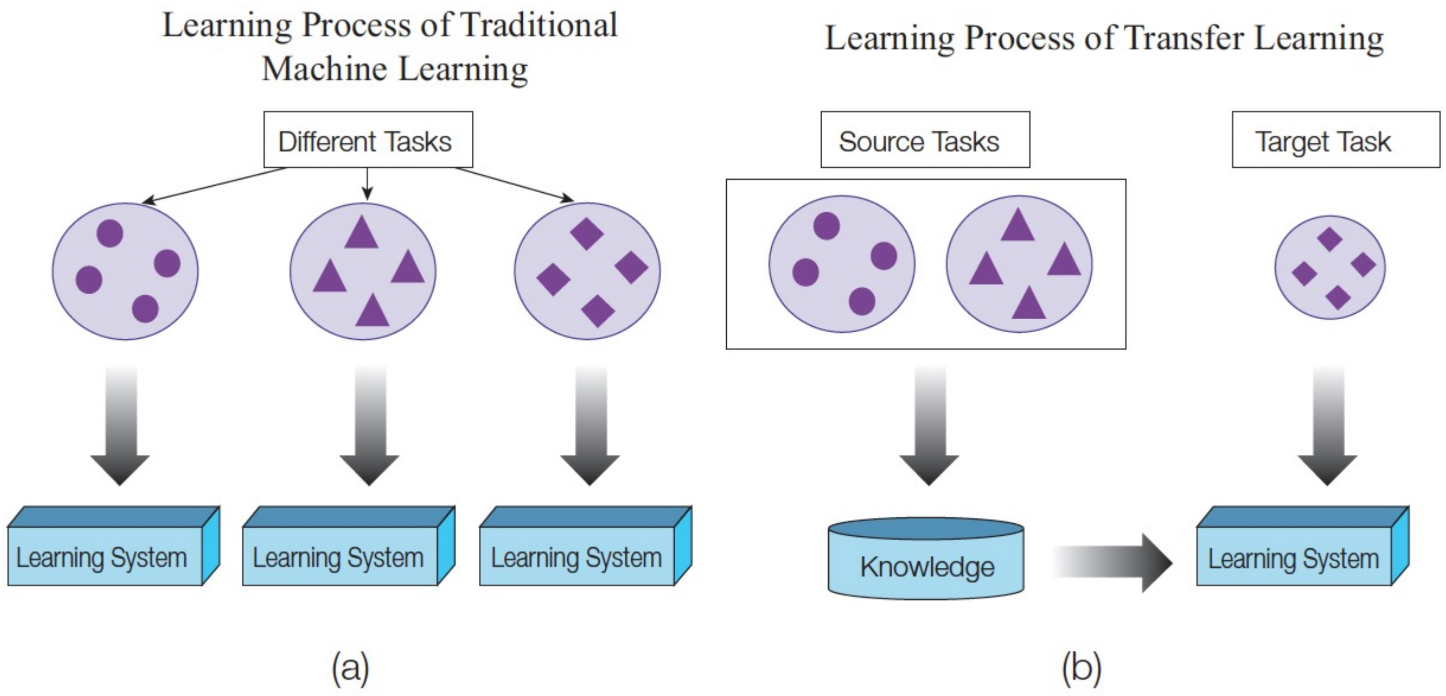


그림 6-5 | 전이 학습의 예

TensorFlow 2.0을 이용한 딥러닝 알고리즘 구현의 2가지 방법

- https://www.tensorflow.org/overview/

For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the "Hello World" example below, then visit the tutorials to learn more.

To learn ML, check out our education page. Begin with curated curriculums to improve your skills in foundational ML areas.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

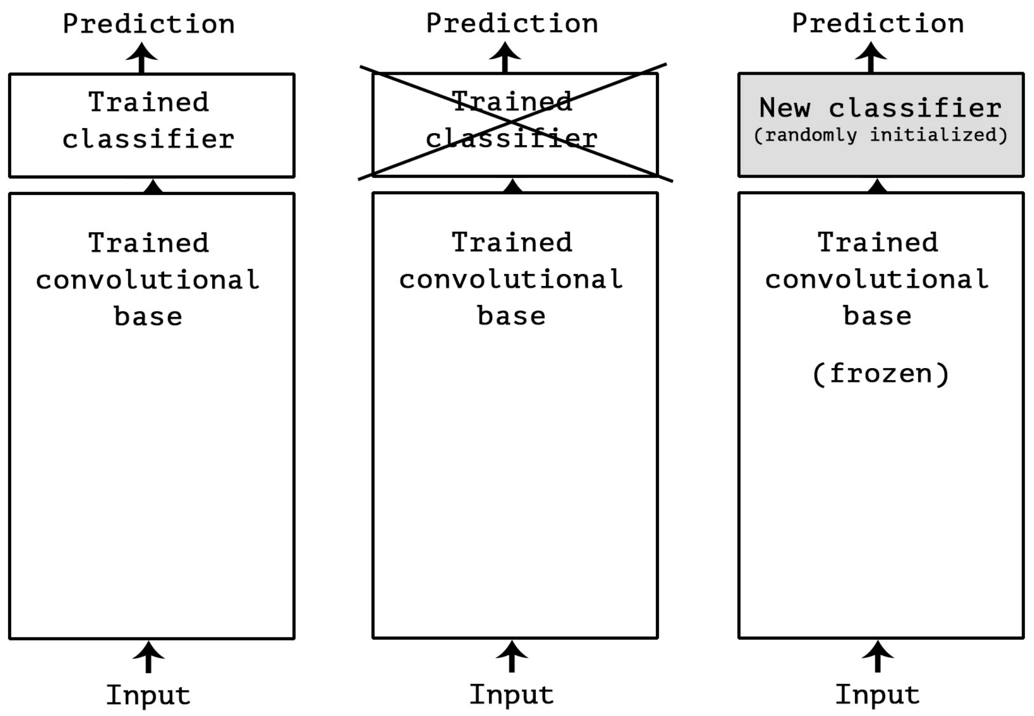
For experts

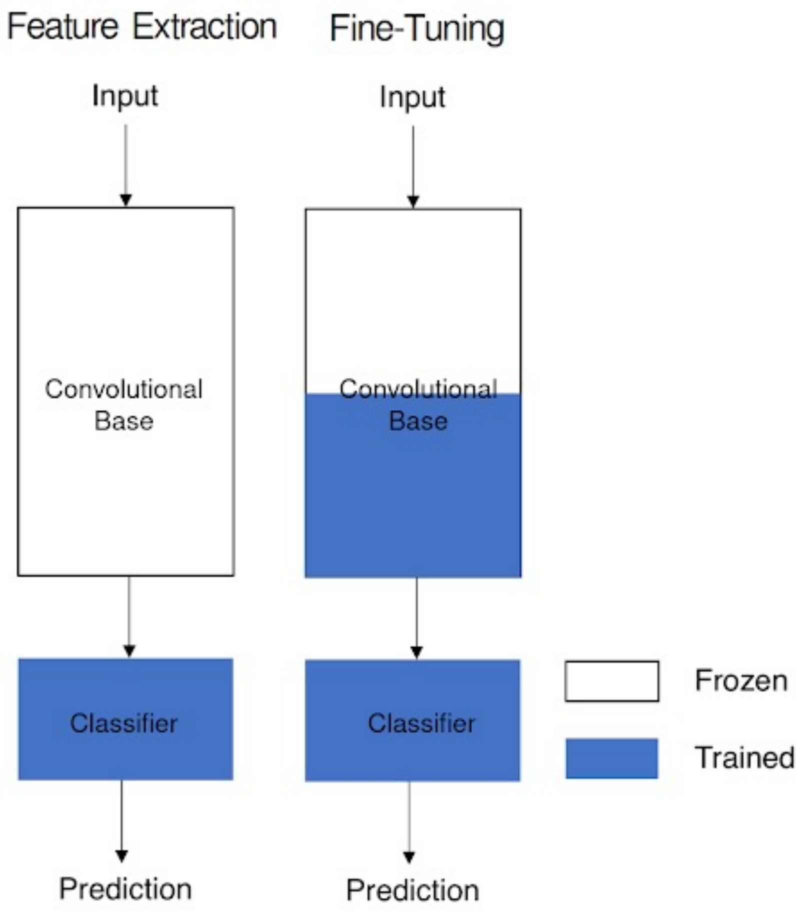
The Subclassing API provides a define-by-run interface for advanced research. Create a class for your model, then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the "Hello World" example below, then visit the tutorials to learn more.

```
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)
grads = tape.gradient(loss_value, model.trainable_variable
optimizer.apply_gradients(zip(grads, model.trainable_varia
```

Pre-Trained CNN을 이용한 Fine-Tuning 과정

- 어디까지를 학습대상으로 삼을지를 결정해서 Fine-Tuning을 진행합니다.





Reference : https://morioh.com/p/bdbd7181cb25

Dogs vs Cats Dataset

- https://www.kaggle.com/c/dogs-vs-cats
- 각각의 25,000장 cat 이미지, 25,000장의 dog 이미지로 구성됨
- Input Image가 cat인지 dog인지를 분류함

tf.keras.applications에서 제공하는 Pre-trained CNN 모델들

- https://keras.io/api/applications/

Available models

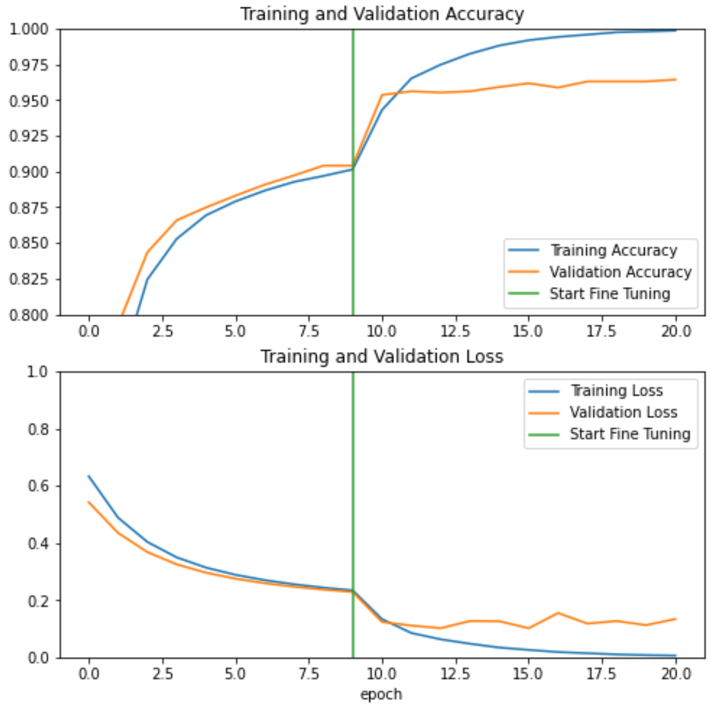
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

VGGNET Fine-Tuning을 통한 Cats vs Dogs 데이터셋 분류

- VGGNET 모델을 이용한 Dogs vs Cats 데이터셋 분류를 위한 Fine-Tuning
- https://colab.research.google.com/drive/1KoeLnZa9wDypT1WQi3Uk5pMy6TLhSTSE?usp=sharing

① Training Softmax Classifier

② Fine-Tuning with last 3 layers



Reference : https://www.tensorflow.org/tutorials/images/transfer_learning

Thank you!