

2021 IoT & AI Vision

1. 인공지능

- 머신러닝
- 이미지 인식(CNN)
- 전이학습

3. TensorFlow Lite

- 이미지 인식(CNN)
- 전이학습
- IoT와 연동

2. 라즈베리파이

- 라즈베리파이 설치
- 원격 접속
- IoT 프로그래밍

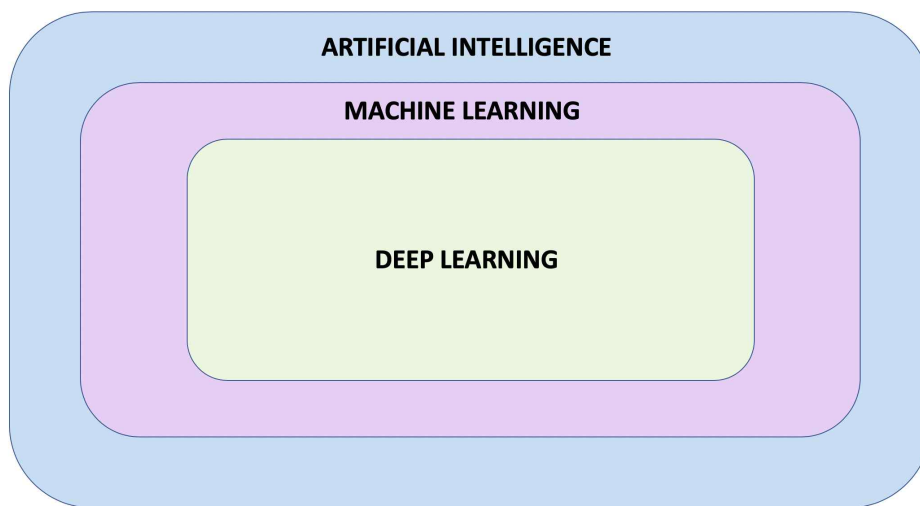
목차

1. 인공지능?
2. 머신러닝 개념
3. 첫 번째 모델 - 패션 MNIST
4. CNN
5. 고양이와 개
6. 전이학습

1. 인공지능?

1) 인공지능, 머신러닝, 신경망, 딥러닝

- **인공 지능:** 컴퓨터가 인간 지능을 달성하도록 하는 것을 목표로 하는 컴퓨터 과학의 한 분야. 머신러닝과 딥러닝을 포함하여 이 목표를 달성하기 위한 많은 접근 방식이 있습니다.
- **머신러닝:** 명시적으로 프로그래밍하지 않고 컴퓨터가 특정 작업을 수행하도록 훈련되는 것과 관련된 기술입니다.
- **신경망:** 생물학적 뇌의 신경망(신경 세포)에서 영감을 받은 머신 러닝의 구조입니다. 신경망은 딥러닝의 기본적인 부분입니다.
- **딥러닝:** 다층 신경망을 사용하는 기계 학습의 하위 분야입니다. 종종 "머신러닝"과 "딥러닝"은 같은 의미로 사용됩니다.



2) 지도 학습, 비지도 학습

지도 학습에서는 컴퓨터에 무엇을 가르치고 싶은지 알고 있는 반면, 비지도 학습은 학습할 수 있는 것을 컴퓨터가 파악하도록 하는 것입니다. 지도학습은 일반적인 유형의 기계 학습이며 이 과정에서 중점적으로 다룰 것입니다.

3) 머신러닝 교육의 네 가지 영역

- **코딩 기술:** ML 모델을 빌드하는 데는 단순한 ML 개념의 이해 이외에도 많은 요소가 개입하며, 데이터 관리, 매개변수 미세 조정, 모델의 테스트와 최적화에 필요한 결과 파싱을 위해 코딩이 필요합니다.
- **수학과 통계:** ML은 수학이 매우 중요하게 사용되는 분야이므로 ML 모델을 수정하거나 새로운 모델을 처음부터 빌드할 계획이라면 기반이 되는 수학 개념에 익숙해야 합니다.
- **ML 이론:** ML 이론의 기초를 알아두면 빌드 작업의 토대가 되고 문제가 발생했을 때 해결하는 데도 도움이 됩니다.
- **나만의 프로젝트:** ML을 직접 경험하는 것이 지식을 습득하는 가장 좋은 방법입니다. 따라서 초기에 간단한 colab이나 튜토리얼을 통해 연습을 시작해 보세요.

4) AI 비전 강의

머신러닝은 몇년 상이에 아주 빠르게 발전해왔습니다. 머신러닝 시스템을 구축하기 위해서 모든 요소들을 직접 만들어야 해서 높은 수준의 수학적 지식이 필요했습니다.

$$J(\Theta) = -\frac{1}{m} \left[\sum_{k=1}^K \sum_{i=1}^m y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

$$\theta_0 := \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \frac{\alpha}{m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda \theta_j \right]; \text{ for } j = 1, \dots, n$$

우리는 고급 인공지능 API를 제공하는 TensorFlow와 Keras를 사용하겠습니다. Python 언어를 사용하고, 실제로 딥러닝 모델을 만들때 Python을 많이 사용합니다. 오늘은 설치 없이 Python을 사용할 수 있는 구글 Colab 환경에서 실습하겠습니다. 수학적 문제나 머신러닝에 대한 깊이 있는 이론 없이 가급적 심플하게 머신러닝에 대해알아보겠습니다.



2. 머신러닝 개념

입력값과 출력값이 다음과 같을때 38을 입력할 때 출력값이 뭐가 될까요?

Input: 0, 8, 15, 22, 38

Output: 32, 46.4, 59, 71.6, ?

다음과 같이 섭씨 온도 C가 주어지면 화씨 온도 F를 구해줍니다.

Input: 0, 8, 15, 22, 38

Output: 32, 46.4, 59, 71.6, ?

$$F = C * 1.8 + 32$$

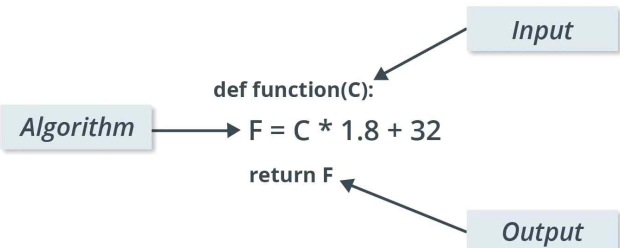
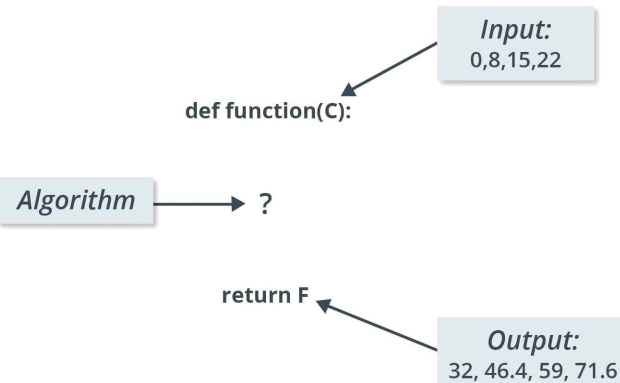
F = 화씨(Fahrenheit)

C = Celsius

Python 함수

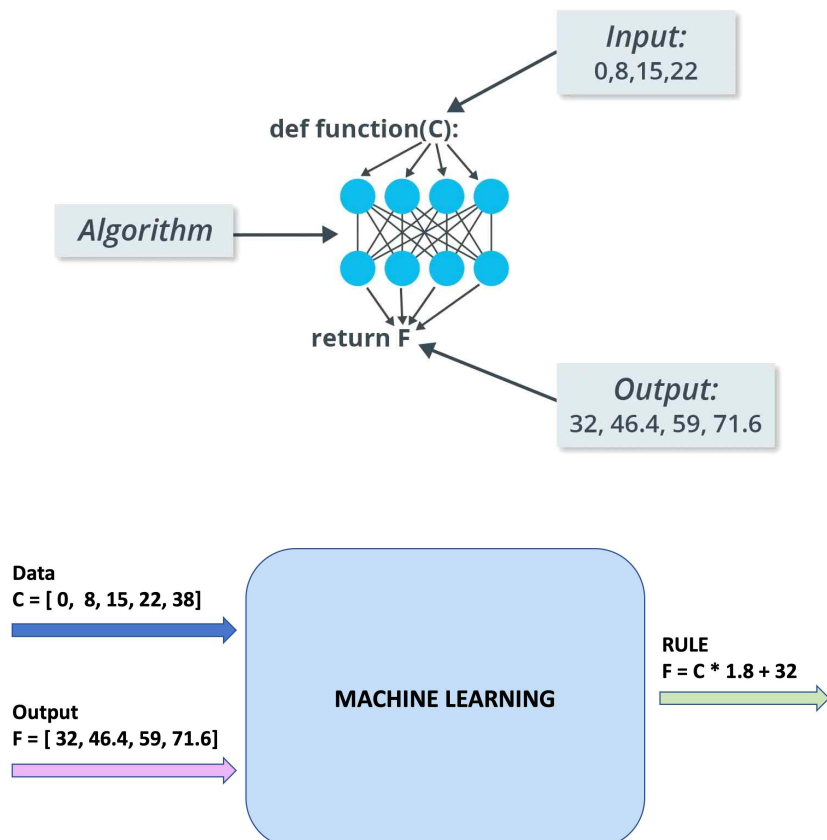
```
def function(C):  
    F = C * 1.8 + 32
```

1) 전통적인 소프트웨어 개발 vs 머신러닝 비교

전통적인 소프트웨어 개발	머신러닝
<p>입력 값과 알고리즘이 주어지면 출력을 위한 함수를 작성합니다.</p> <ul style="list-style-type: none"> - 입력 데이터 - 입력데이터를 계산식에 적용 - 결과값 리턴 	<p>입력값과 출력값을 알고있지만 알고리즘을 모르는 상태입니다.</p> <ul style="list-style-type: none"> - 입력값과 출력값의 쌍을 입력 - 알고리즘 찾기
	

2) 머신러닝

신경망은 층들의 집합이라 볼수 있습니다. 각층은 내부 변수값을 가지고 있고 입력값과 변수값에 수학적 연산을 하여 다음 층으로 보냅니다. 입력값과 출력값으로 내부 변수를 튜닝하면서 수천~수만번 반복하면서 학습합니다.



3) 머신러닝 용어

- 모델(Model) - 기계학습 알고리즘.
- 특징(Feature) - 모델에 대한 입력입니다. 섭씨온도
- 레이블(Labels) - 모델이 예측하는 출력입니다. 화씨온도
- Example - 훈련 중에 사용된 한 쌍의 입력/출력 값의 쌍(22, 72).

4) TensorFlow 화씨 온도 변환

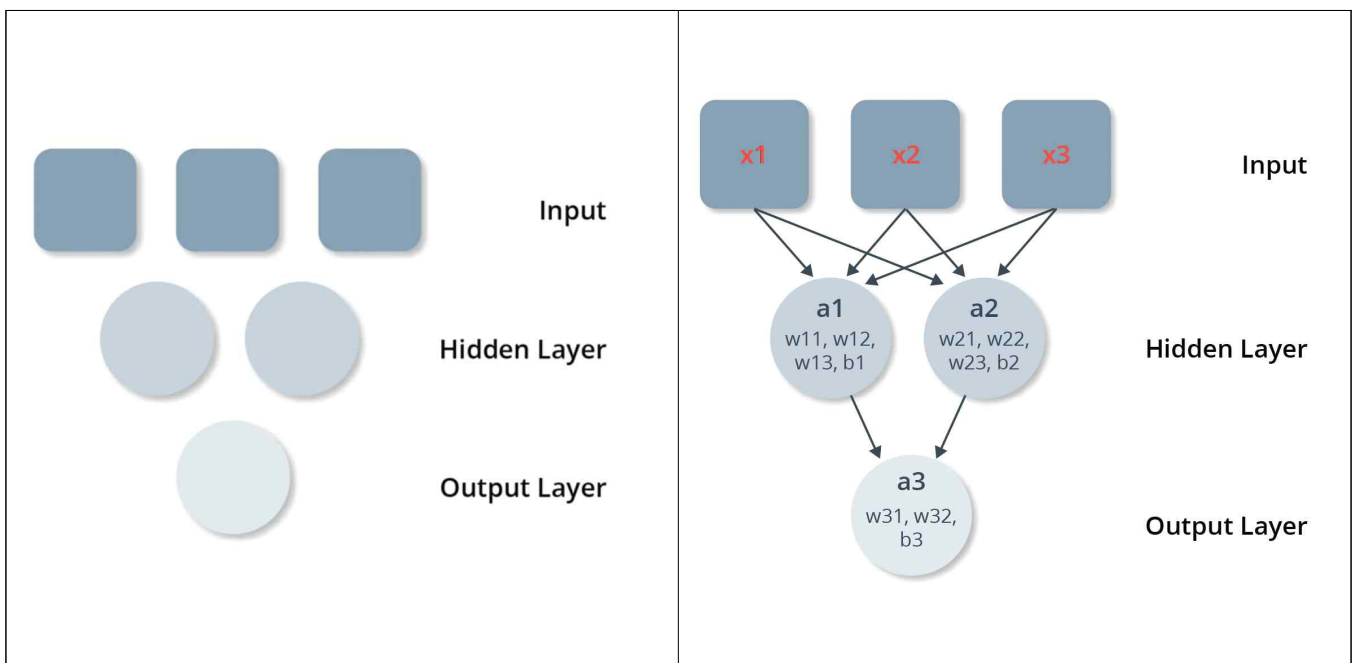
```
l0 = tf.keras.layers.Dense(units=1, input_shape=[1])  
model = tf.keras.Sequential([l0])  
model.compile(loss='mean_squared_error', optimizer=tf.keras.optimizers.Adam(0.1))  
history = model.fit(celsius_q, fahrenheit_a, epochs=500, verbose=False)  
model.predict([100.0])
```

5) Dense Layers

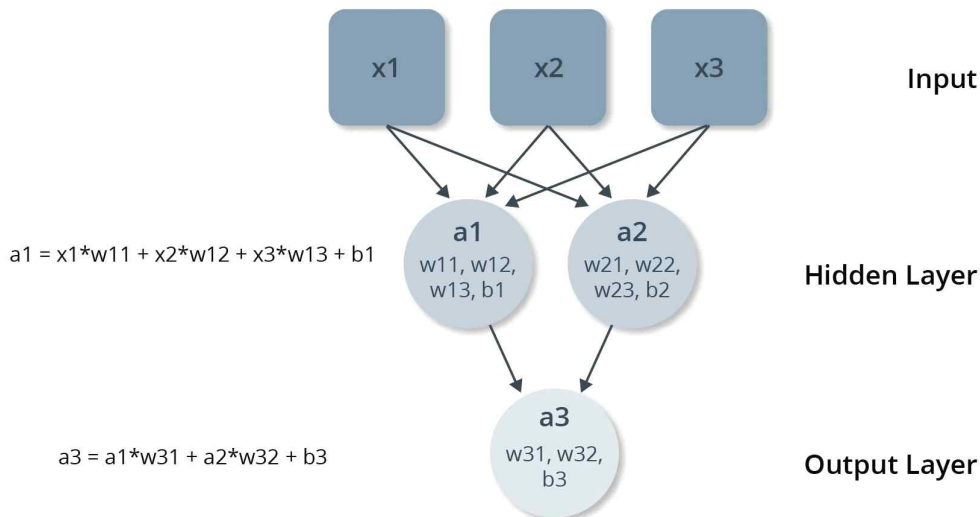
- Python

```
l0 = tf.keras.layers.Dense(units=1, input_shape=[1])  
  
model = tf.keras.Sequential([l0])
```

- 입력 3개, 히든 2개, 출력 1개

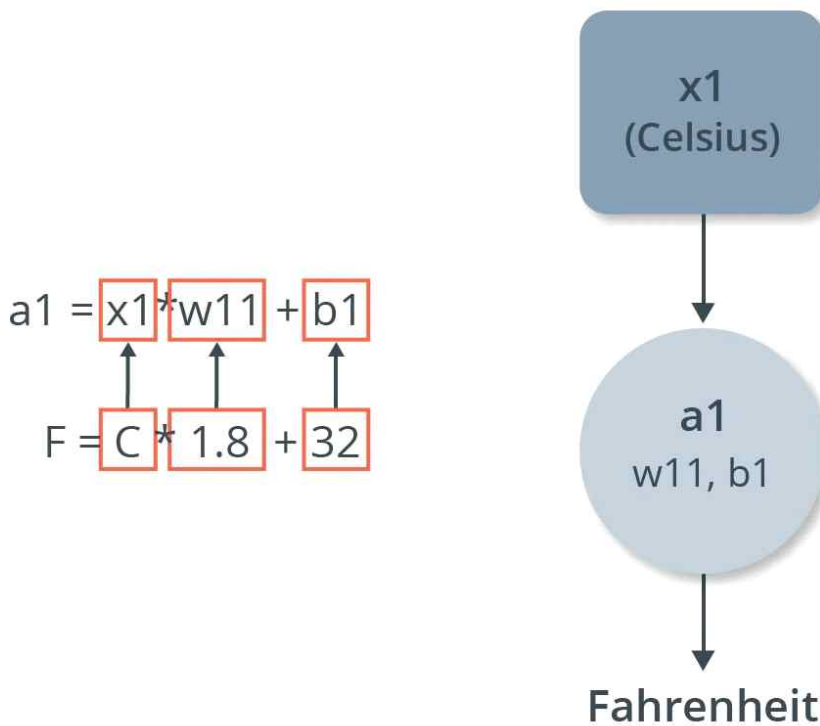


· 가중치(weight), 편향(bias)



5) 섭씨를 화씨로 변환

```
[ ] l0 = tf.keras.layers.Dense(units=1, input_shape=[1])
```



6) 내부 변수

```
[ ] print("These are the layer variables: {}".format(l0.get_weights()))
```

These are the layer variables: [array([[1.8215632]], dtype=float32), array([29.174837], dtype=float32)]


```
[20] l0 = tf.keras.layers.Dense(units=4, input_shape=[1])
    l1 = tf.keras.layers.Dense(units=4)
    l2 = tf.keras.layers.Dense(units=1)
    model = tf.keras.Sequential([l0,l1,l2])
    model.compile(loss='mean_squared_error',optimizer=tf.keras.optimizer.Adam(0.1))
    model.fit(celsius_q, fahrenheit_a, epochs=500, verbose=False)
    print("Finished training the model")
    print(model.predict([100.0]))
    print("Model predicts that 100 degrees Celsius is: {} degrees Fahrenheit".format(model.predict([100.00])))
    print("These are the l0 variables: {}".format(l0.get_weights()))
    print("These are the l1 variables: {}".format(l1.get_weights()))
    print("These are the l2 variables: {}".format(l2.get_weights()))
```

```
Finished training the model
[[211.74742]]
Model predicts that 100 degrees Celsius is: [[211.74742]] degrees Fahrenheit
These are the l0 variables: [array([[ -0.18800685, -0.19248432, -0.5858944, -0.67664355]],
      dtype=float32), array([ -3.5765584,  1.7353412, -3.6145658, -3.606096 ], dtype=float32)]
These are the l1 variables: [array([[ 1.0775942,  0.3412887, -0.20648848,  1.0967993 ],
      [ 0.38938883, -0.01514232,  0.62254876, -1.0326661 ],
      [ 0.34104902, -0.23185489, -0.68317974, -0.02780363],
      [ 0.5441887, -0.2788269, -0.5909913,  0.32856533]],
      dtype=float32), array([ -3.5302405,  1.1981959,  3.5821548, -3.5761232], dtype=float32)]
These are the l2 variables: [array([[-1.0786525],
      [ 0.20307049],
      [ 0.92228246],
      [-0.7923224]], dtype=float32), array([3.4605324], dtype=float32)]
```

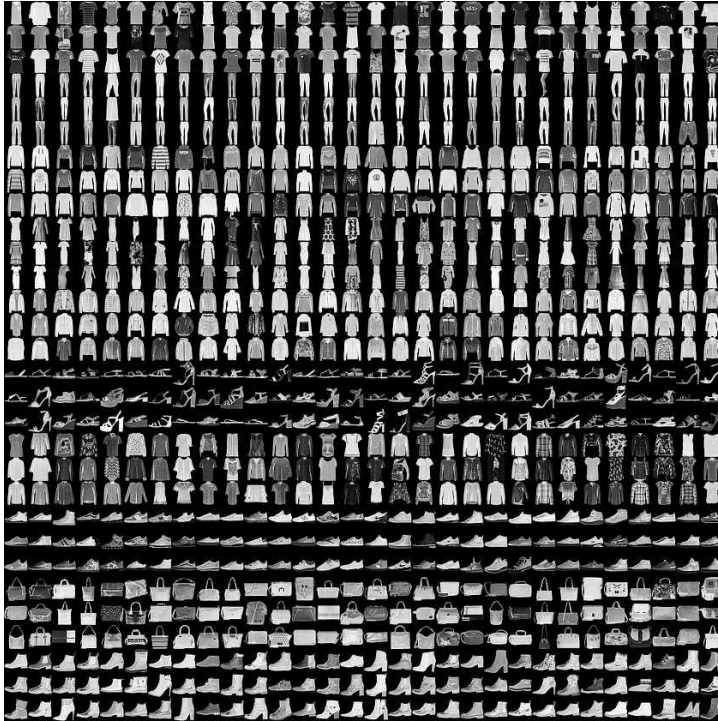
7) 요약

이 단원에서는 기계 학습의 기본 개념과 Dense 레이어의 작동 방식에 대해 배웠습니다. 섭씨를 화씨로 변환하도록 첫 번째 기계 학습 모델을 훈련했습니다. 또한 기능, 예제 및 레이블과 같이 기계 학습에서 사용되는 몇 가지 주요 용어를 배웠습니다. 또한 모든 기계 학습 알고리즘의 골격을 구성하는 주요 코드 라인을 배웠습니다. TensorFlow 및 Keras를 사용하여 신경망으로 생성, 교육 및 예측을 수행하는 데 몇 줄의 코드만 있으면 됩니다.

3. 첫 번째 모델 - 패션 MNIST

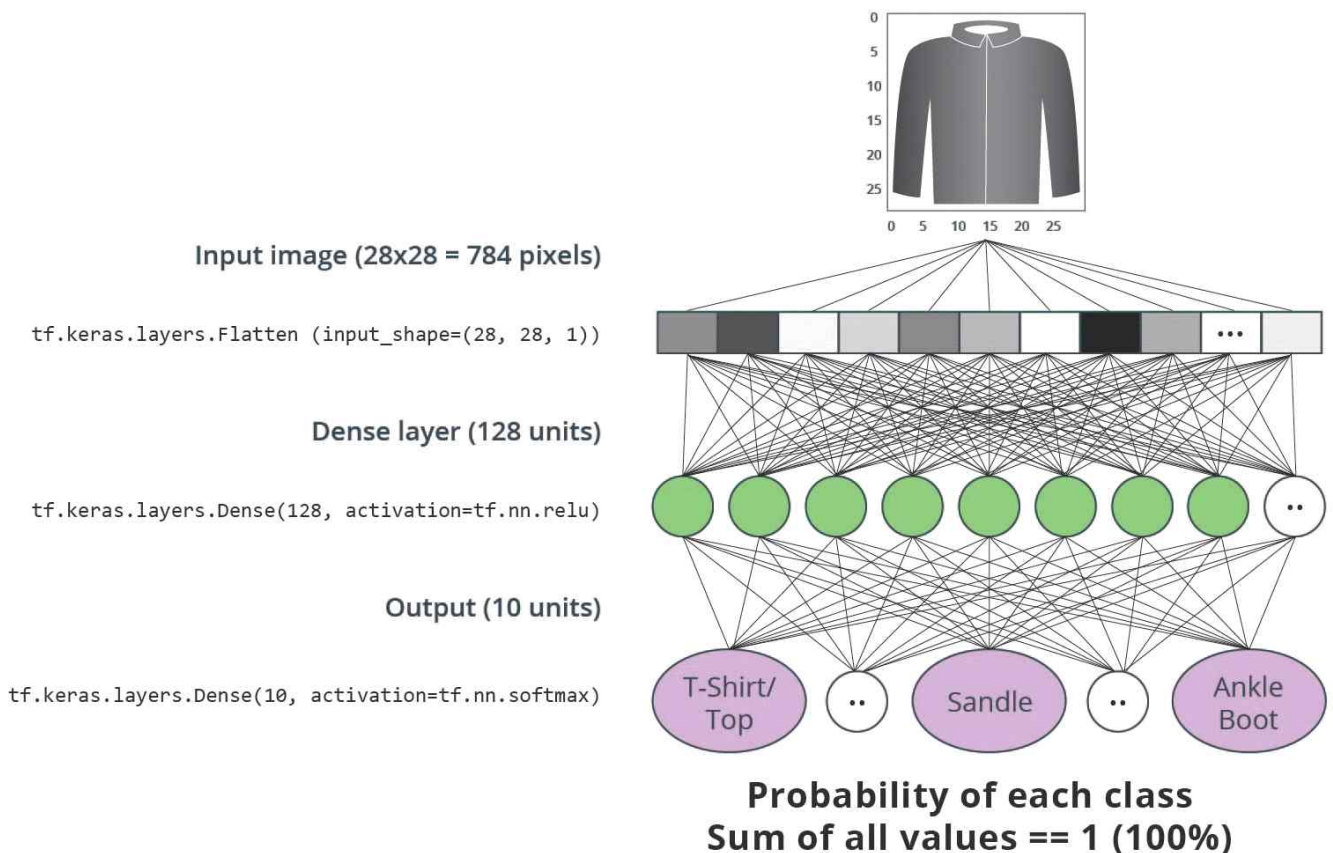
1) 패션 MNIST 데이터셋

패션-MNIST는 60,000개의 예제로 구성된 훈련세트와 10,000개의 예제로 구성된 테스트세트로 구성된 Zalando 의 의류 이미지 데이터셋입니다. 각 예제는 10개 클래스의 레이블과 연결된 28x28 회색조 이미지입니다. 각 이미지의 크기는 784바이트 입니다.



Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

2) 패션 MNIST 신경망

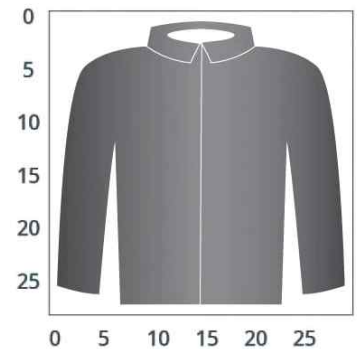


3) 결과

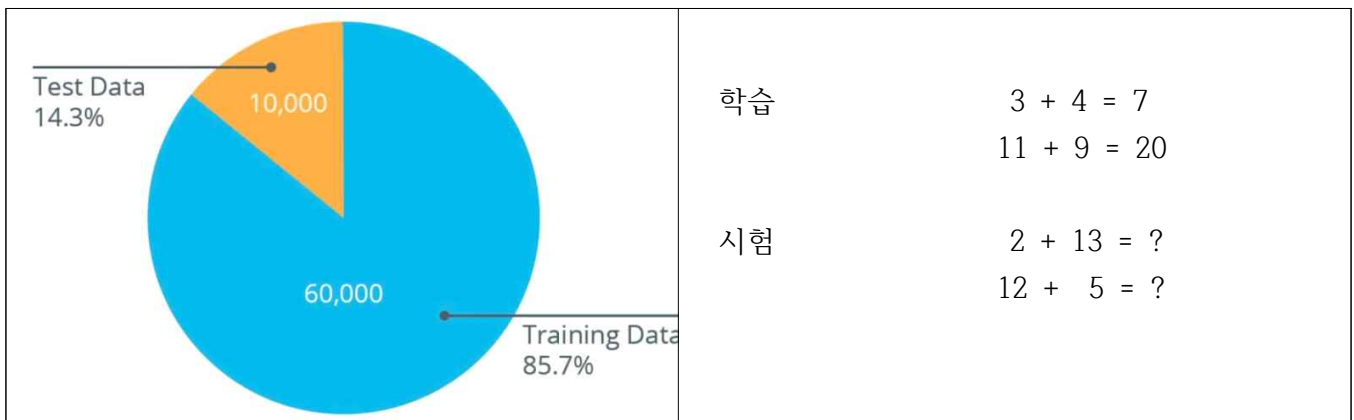
Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

0.02
0.01
0.03
0.01
0.05
0.01
0.85
0.006
0.007
0.007

Sum to 1



4) 훈련 및 테스트



훈련 데이터란 머신 러닝에서 학습을 할 데이터입니다. 훈련 데이터는 훈련 데이터셋이라고도 불립니다. 이런 훈련 데이터를 학습하여 모델을 만들고, 훈련 데이터에 따라서 만들어지는 모델이 달라집니다. 사람과 마찬가지로 문제들을 많이 푼다고해서 즉, 훈련 데이터를 많이 학습한다고 해서 결과가 잘 나오지는 않습니다.

시험에 필요한 데이터가 테스트 데이터입니다. 테스트 데이터셋이라고도 불립니다. 학습 과정에서 문제를 통째로 외워버릴 수도 있기 때문에 테스트 데이터가 필요합니다. 훈련 데이터와 테스트 데이터는 겹치지 않게 해야됩니다.

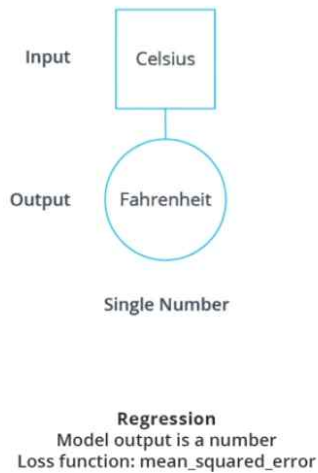
과대적합은 모델의 훈련 세트 점수가 테스트 세트 점수보다 훨씬 높을 경우를 의미합니다.

과소적합은 이와 반대로 모델의 훈련 세트와 테스트 세트 점수가 모두 동일하게 낮거나 테스트 세트 성능이 오히려 더 높을 경우를 의미합니다.

5) Colab: 패션 MNIST

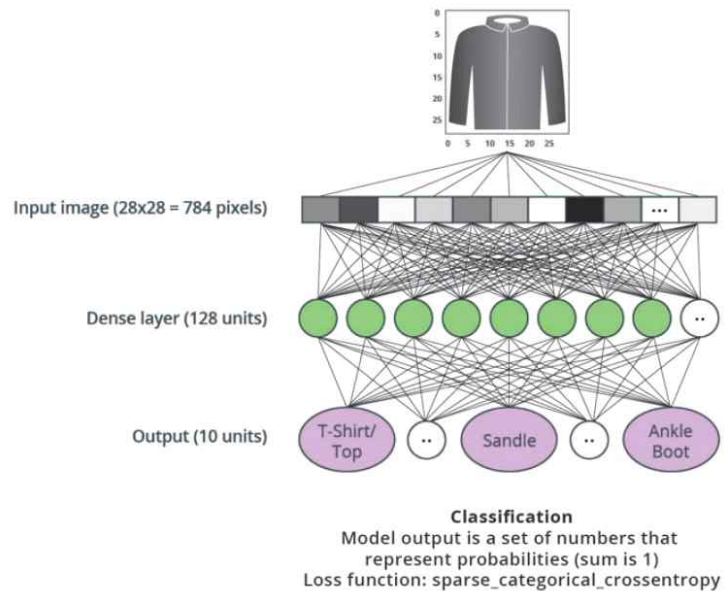
6) 섭씨 온도 vs 패션 MNIST 데이터 셋

Celsius to Fahrenheit



<회귀문제>

Fashion MNIST



<분류 문제>

7) 회귀(Regression) vs 분류(Classification)

- 회귀: 단일 값을 출력하는 모델입니다. 예를 들어, 주택 가치의 추정.
- 분류: 여러 범주에 걸쳐 확률 분포를 출력하는 모델입니다. 예를 들어, Fashion MNIST에서 출력은 10개의 확률이었고, 각각 다른 유형의 의복에 대해 하나씩입니다. 이 확률 분포를 생성하기 위해 마지막 Dense 레이어에서 활성화 함수로 Softmax를 사용한다는 것을 기억하십시오.

	Classification	Regression
Output	List of numbers that represent probabilities for each class	Single Number
Example	Fashin MNIST	Celsius to Fahrenheit
Loss	Sparse categorical crossentropy	Mean squared error
Last Layer Activation Function	Softmax	None

옷의 이미지를 분류하기 위해 신경망을 훈련했습니다. 이를 위해 70,000개의 회색조 의류 이미지가 포함된 Fashion MNIST 데이터 세트를 사용했습니다. 그 중 60,000개는 네트워크를 훈련하는 데 사용하고 10,000개는 성능을 테스트하는 데 사용했습니다. 이 이미지를 신경망에 제공하기 위해 28×28 이미지를 784개 요소가 있는 1d 벡터로 평면화해야 했습니다. 우리의 네트워크는 128개 단위(뉴런)가 있는 완전 연결 계층과 10개의 출력 레이블에 해당하는 10개 단위가 있는 출력 계층으로 구성됩니다. 이 10개의 출력은 각 클래스에 대한 확률을 나타냅니다. softmax 활성화 함수는 확률 분포를 계산했습니다.

4. CNN

CNN: 컨볼루션 신경망. 적어도 하나의 컨볼루션 레이어가 있는 네트워크입니다. 일반적인 CNN에는 풀링 레이어 및 Dense 레이어와 같은 다른 유형의 레이어도 포함됩니다.

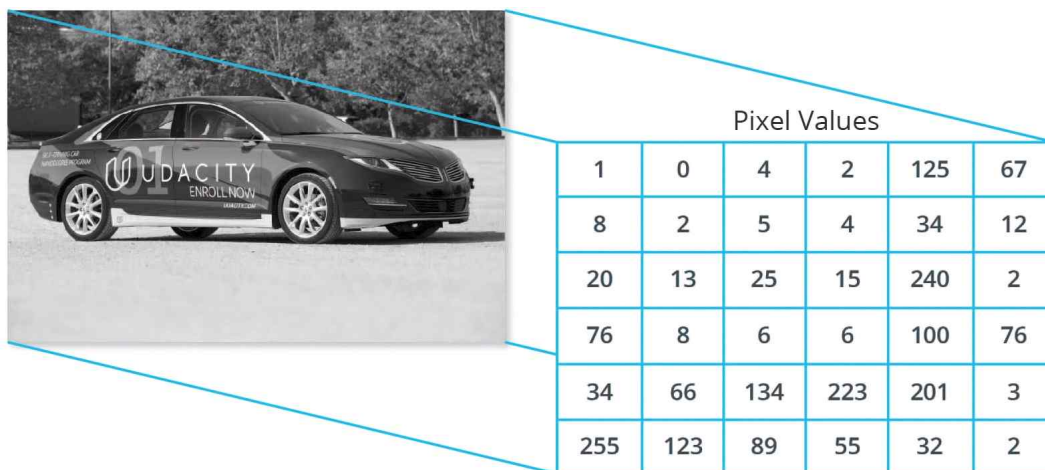
- Convolutions
- MaxPooling

1) 컨볼루션

Greyscale Image



그레이스케일 이미지가 6 x 6 픽셀로 이루어져 있다고 가정합니다. 그레이스케일이기 때문에 각 픽셀은 0~255 상이의 값을 가집니다. 0은 검정색이고 255는 하얀색입니다.



컨볼루션은 이미지에 필터("커널")를 적용하는 과정입니다. 맥스 풀링은 다운샘플링을 통해 이미지의 크기를 줄이는 과정입니다.

Keras의 Conv2D레이어 유형을 사용하여 신경망 모델에 컨볼루션 레이어를 추가할 수 있습니다. 이 레이어는 Dense 레이어와 유사하며 가중치와 편향이 있습니다. 이 Conv2D 계층에는 값을 조정해야 하는 커널(필터)도 있습니다. 따라서 Conv2D 레이어에서 필터 내부의 값은 올바른 출력을 생성하기 위한 변수입니다.

커널 / 필터: 입력보다 작은 행렬로 입력을 청크로 변환하는 데 사용됩니다.

Pixel Values

1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2

Kernel 3 x 3 Pixels

1	2	1
2	4	2
1	2	1

각 이미지 사각형 영역에 각 픽셀 값과 커널 영역의 대응하는 영역의 값을 곱한 후 전체 값을 더합니다.

Pixel Values

1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2

Kernel 3 x 3 Pixels

1	2	1
2	4	2
1	2	1

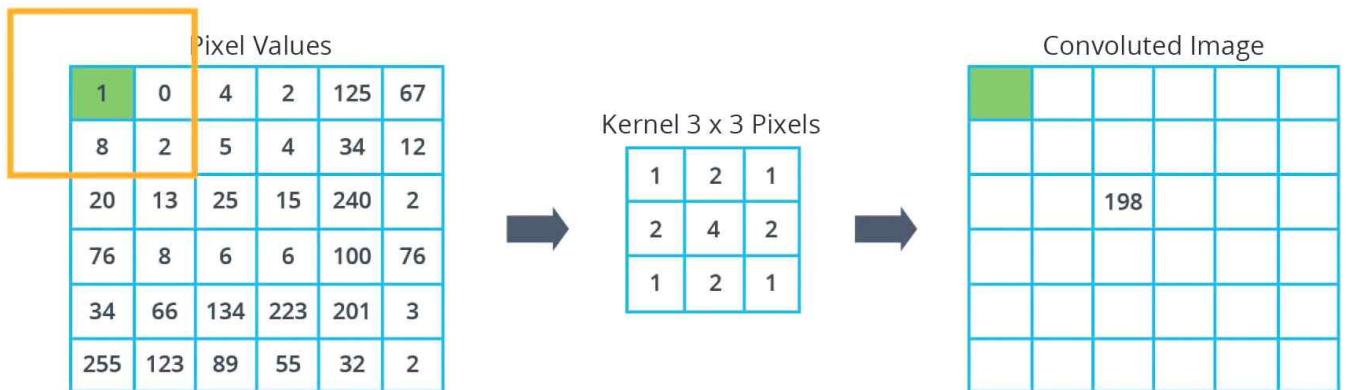
Convolved Image

		198			

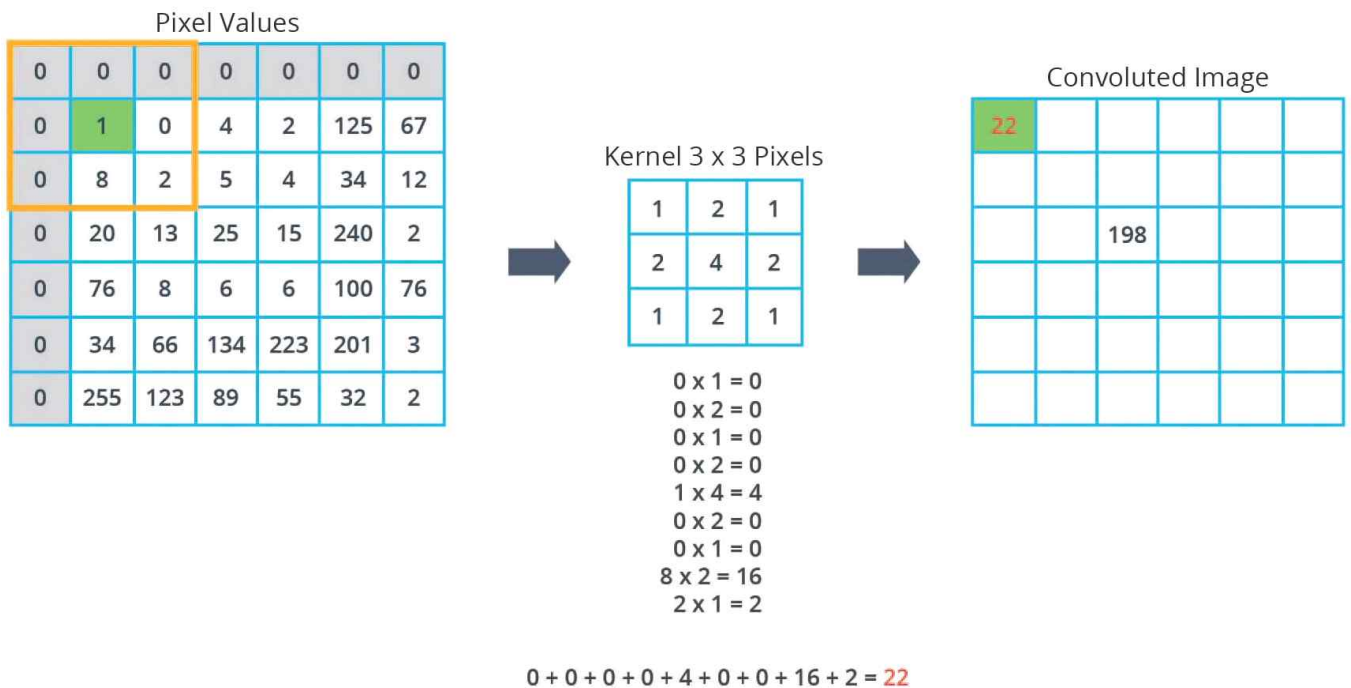
$$\begin{aligned}
 2 \times 1 &= 2 \\
 2 \times 5 &= 10 \\
 1 \times 4 &= 4 \\
 2 \times 13 &= 26 \\
 25 \times 4 &= 100 \\
 15 \times 2 &= 30 \\
 8 \times 1 &= 8 \\
 6 \times 2 &= 12 \\
 6 \times 1 &= 6
 \end{aligned}$$

$$2 + 10 + 4 + 26 + 100 + 30 + 8 + 12 + 6 = 198$$

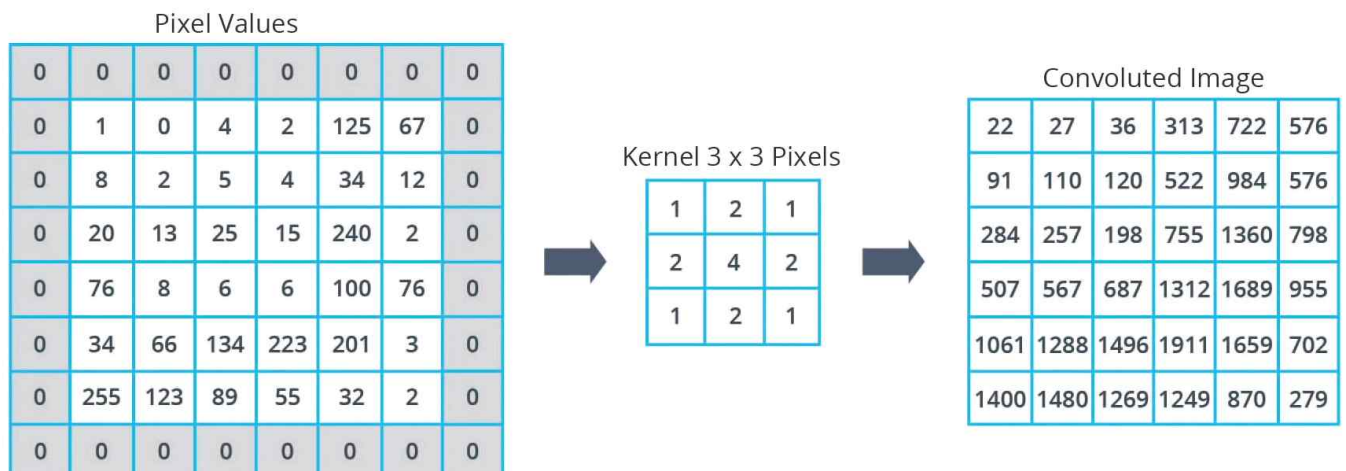
가장자리에 있는 픽셀의 값을 구하는 방법을 알아보겠습니다. 이런 경우 첫번째 픽셀을 무시하는 방법도 있습니다. 이경우 이미지 사이즈가 다운사이즈 됩니다. 다른 방법은 0 패딩 입니다. 원래 이미지 주변에 0을 더하면 됩니다. 그럼 가장자리 픽셀도 다른 영역의 픽셀 처럼 계산 할 수 있습니다.



첫번째 픽셀에 대해서 계산해 보겠습니다.



전체영역에 대해 계산해보면 다음과 같습니다.



따라서 커널 컨볼루션은 입력 이미지에 커널 또는 필터를 적용하는 과정입니다. Dense 층과 같이 컨볼루션은 케라스 층의 또 다른 타입입니다.

2) 최대풀링(Max-Pooling)

패딩: 입력 이미지 주위에 어떤 값(보통 0)의 픽셀 추가

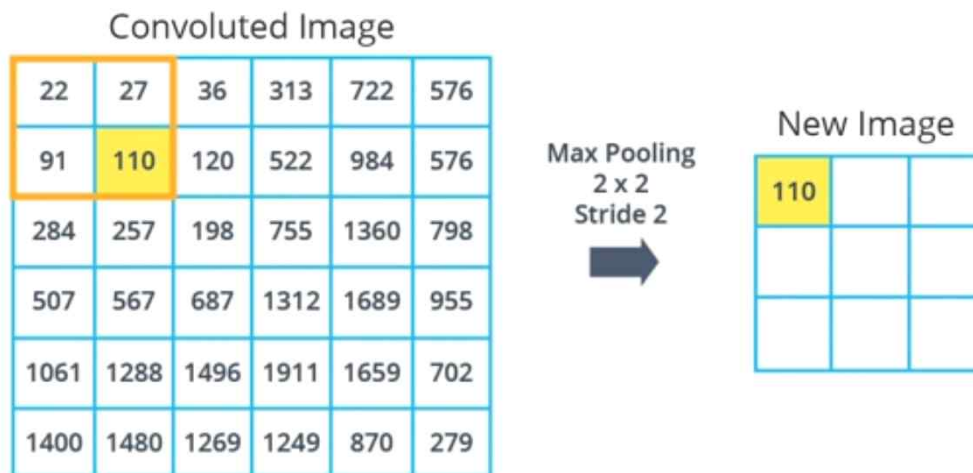
풀링: 다운샘플링을 통해 이미지의 크기를 줄이는 과정입니다. 풀링 레이어에는 여러 유형이 있습니다. 예를 들어, 평균 풀링은 평균을 취하여 많은 값을 단일 값으로 변환합니다. 그러나 maxpooling이 가장 일반적입니다.

Maxpooling : 많은 값 중에서 최대값을 취하여 하나의 값으로 변환하는 pooling 과정.

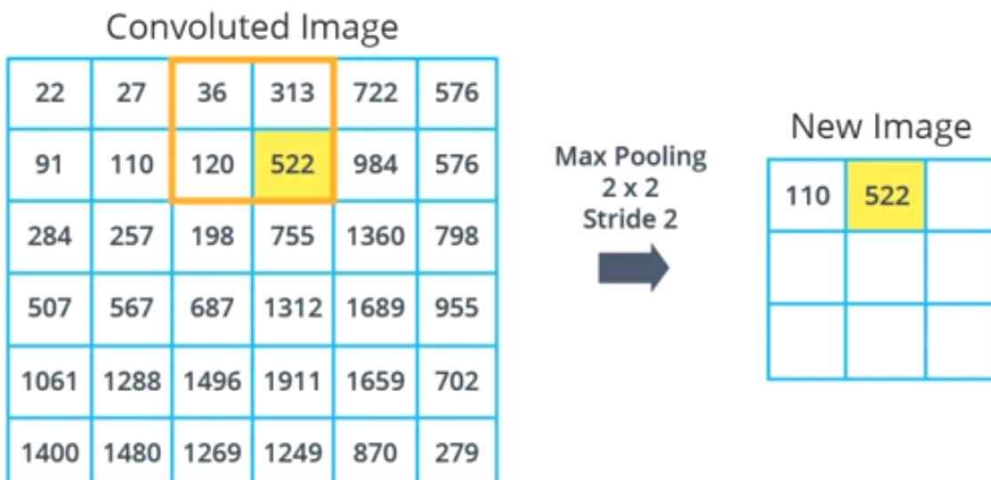
Stride: 이미지에서 커널(필터)을 슬라이드할 픽셀 수입니다.

다운샘플링: 이미지의 크기를 줄이는 행위

풀링



스트라이트



Convolved Image

22	27	36	313	722	576
91	110	120	522	984	576
284	257	198	755	1360	798
507	567	687	1312	1689	955
1061	1288	1496	1911	1659	702
1400	1480	1269	1249	870	279

Max Pooling
2 x 2
Stride 2



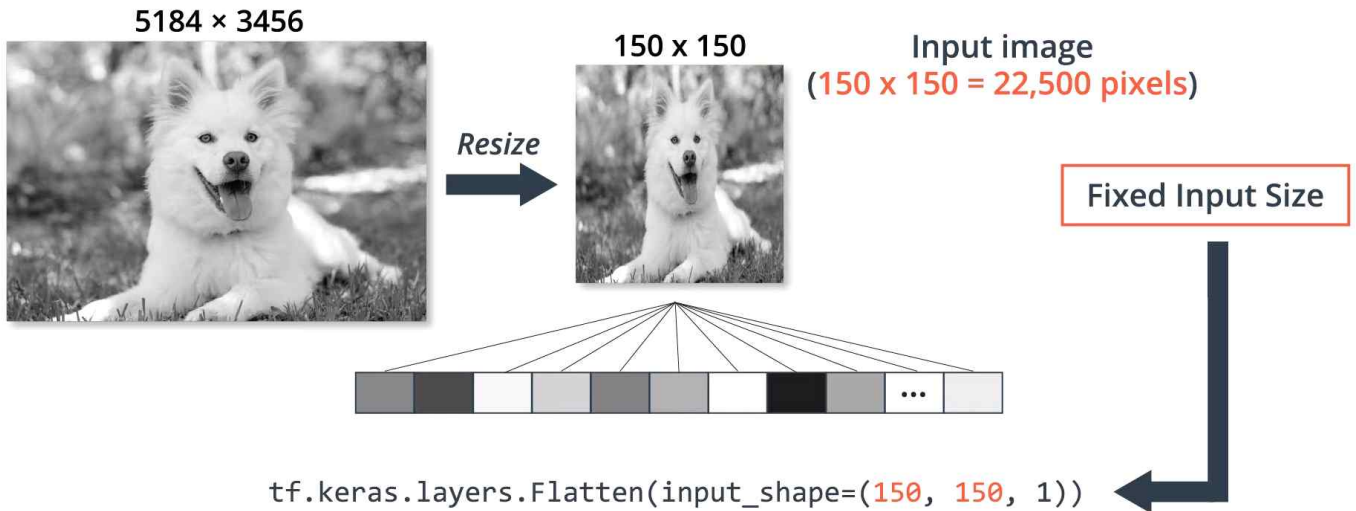
New Image

110	522	984
567	1312	1689
1480	1911	1659

3) CNN과 함께하는 패션 MNIST

5. 고양이와 개

1) 이미지 크기 조정



2) Colab 고양이와 개

훈련 세트에서는 정확도가 100%나오지만, 검증세트에서는 정확도가 75%입니다.

3) 소프트맥스 시그모이드

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])
```

마지막 레이어(분류기)는 출력 클래스 수와 활성화함수가 있는 Dense 레이어로 구성됩니다.

```
tf.keras.layers.Dense(2, activation='softmax')
```

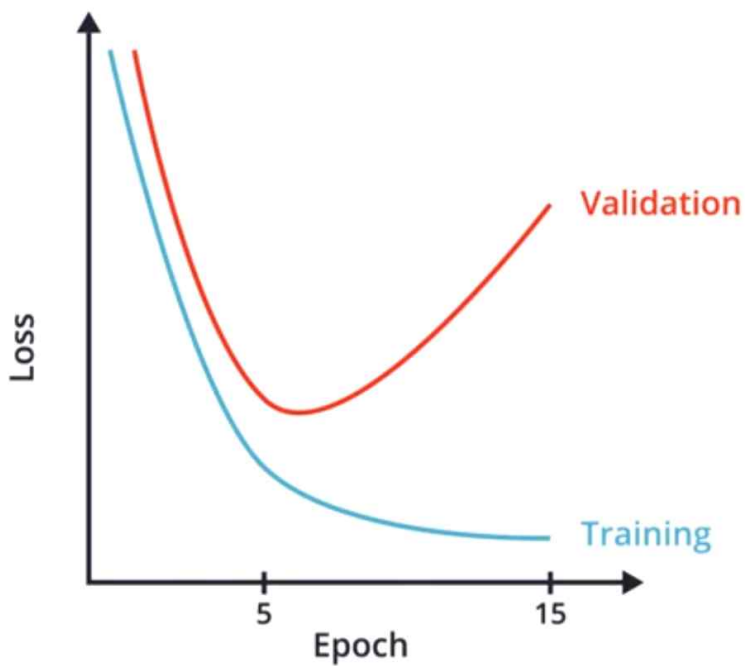
이진 분류 문제로 작업할 때 널리 사용되는 방식은 다음과 같습니다. 이 방식은 이진 분류 문제에서 잘 동작합니다.

```
tf.keras.layers.Dense(1, activation='sigmoid')
```

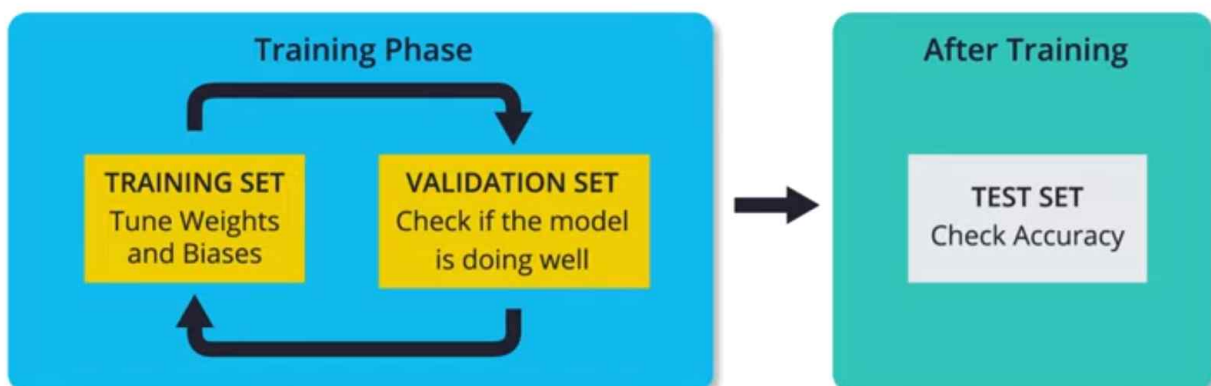
sigmoid함수를 사용하는 경우 model.compile() 메서드에서 loss 함수를 다음과 같이 'sparse_categorical_crossentropy'에서 'binary_crossentropy'로 수정해야 합니다.

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

4) 검증

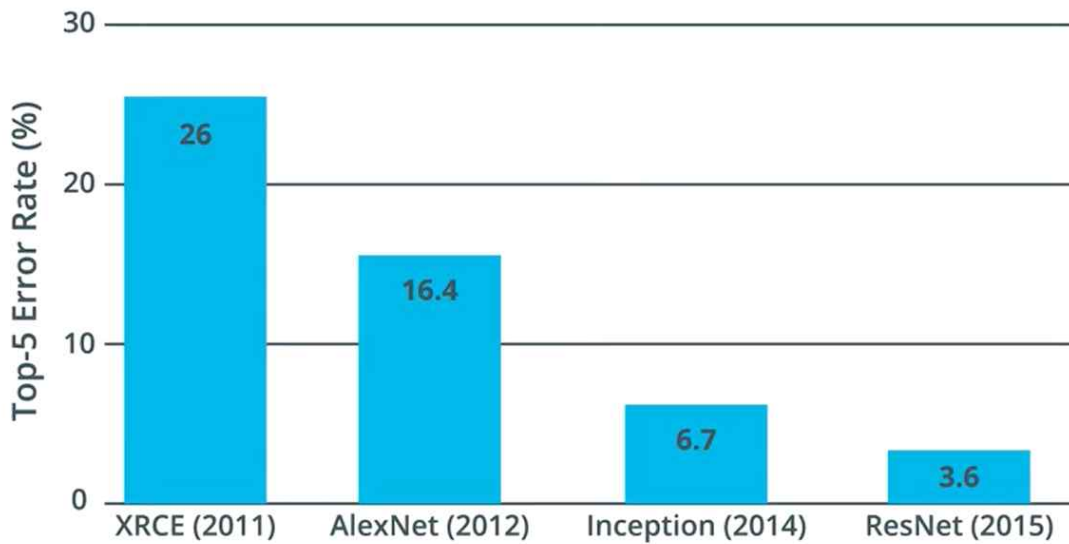


5) 훈련세트, 검증세트, 테스트 세트



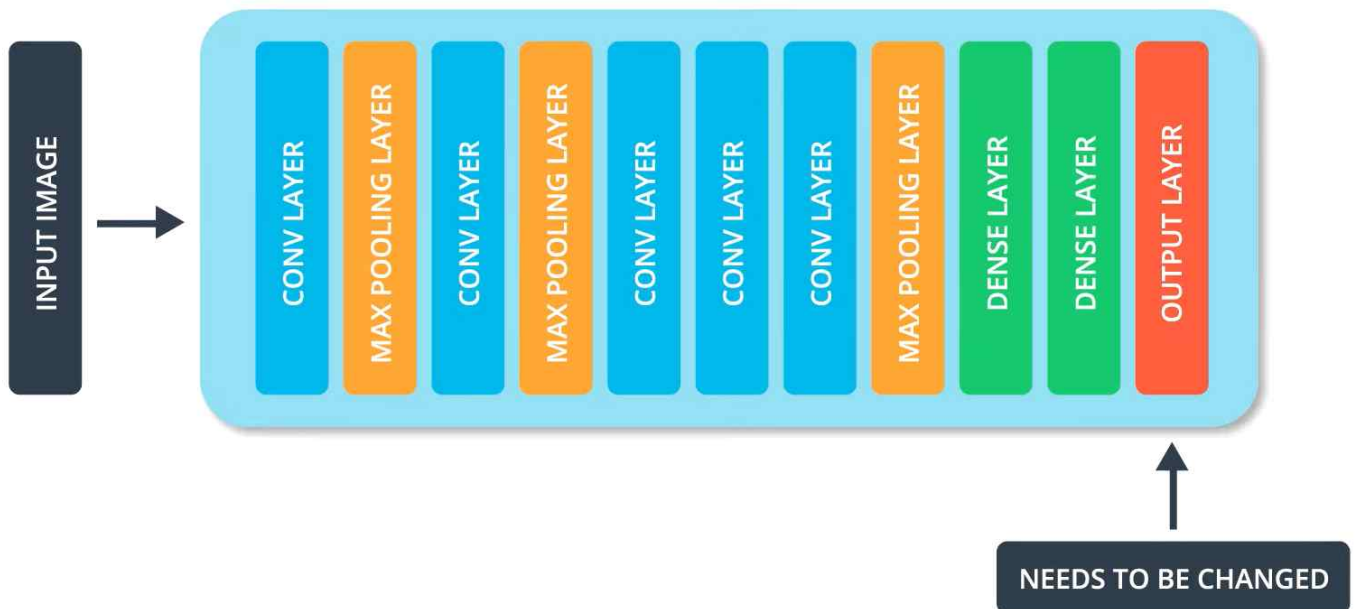
6. 전이학습

IMAGENET



전이학습을 대규모 데이터 세트에서 훈련된 신경망이 새로운 데이터 세트에 적용할 수 있습니다. 전이 학습은 이미 학습된 모델을 이용하여 새로운 데이터셋 학습에 사용할 수 있습니다.

PRE-TRAINED MODEL



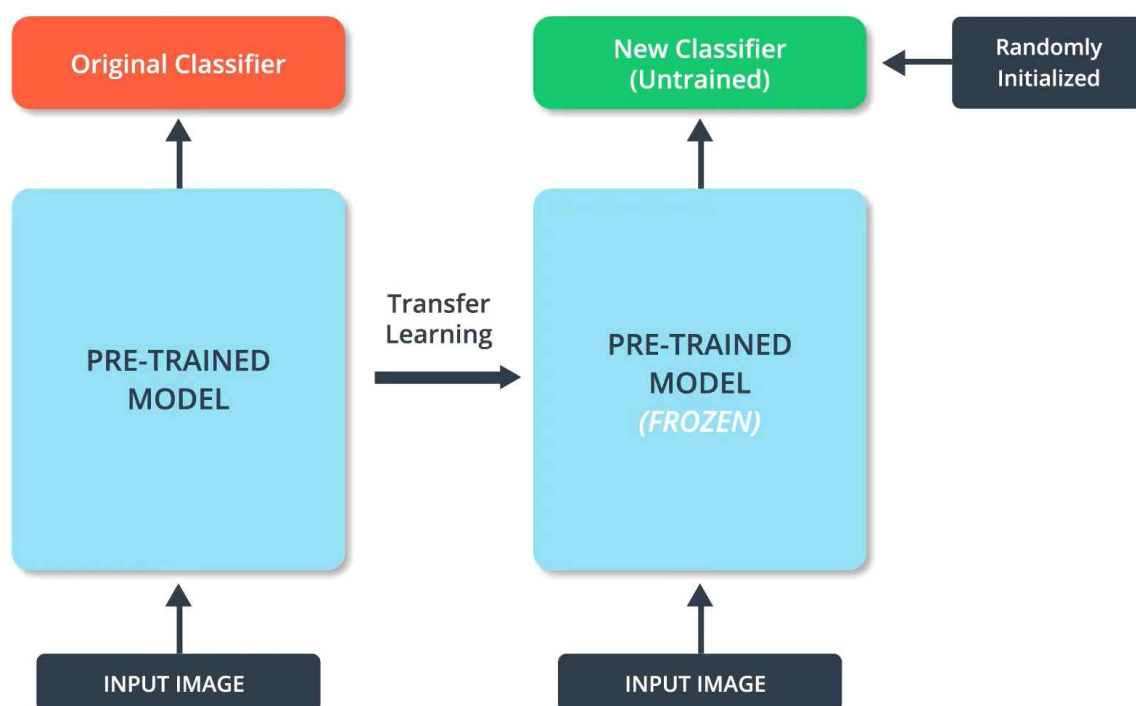
전이학습을 위해서는 이미 학습된 모델의 마지막 층을 제거 해야 합니다.

Dataset	Output Classes
ImageNet	1000
Fashion MNIST	10
Cats and Dogs	2

데이터셋들 마다 클래스 수가 달라서 출력 Dense Layer를 수정해야 합니다.



이미 학습된 모델의 변수들은 학습할 수 없도록 합니다. 마지막 층의 변수만 학습될 수 있도록 합니다. 전이 학습은 마지막 층만 학습하기 때문에 학습 시간도 절약됩니다.



7. 모바일넷

앞에서 언급된 AlexNet과 ResNet은 정확성은 높지만 많은 변수들을 사용합니다. 그래서 예측을 위해 많은 메모리를 사용해야 하고 많은 복잡한 연산을 해야 합니다. 이번 장에서는 구글서 만들 MobileNet에 대해 알아보겠습니다. MobileNet은 높은 정확성을 유지하면서도 메모리 사용량이 적고 낮은 사양의 컴퓨터에서도 실행할 수 있습니다.

MobileNet을 다운로드 하겠습니다. MobileNet 은 224 * 224, RGB 이미지로 학습되어 사이즈를 고정해줍니다.

```
CLASSIFIER_URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/2"
IMAGE_RES = 224
model = tf.keras.Sequential([
    hub.KerasLayer(CLASSIFIER_URL, input_shape=(IMAGE_RES, IMAGE_RES, 3))
])
```

TensorFlow hub는 이미 훈련된 모델의 저장소 입니다. TensorFlow hub의 모델은 마지막 층이 제거되었기 때문에 클래스 수에 맞는 classification layer(마지막층)를 추가하기만 하면 됩니다.

```
CLASSIFIER_URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/2"
IMAGE_RES = 224
model = tf.keras.Sequential([
    hub.KerasLayer(CLASSIFIER_URL, input_shape=(IMAGE_RES, IMAGE_RES, 3))
    layers.Dense(2, activation='softmax')
])
```

이전에 사용하던 fit() 메서드를 이용하여 모델을 학습할 수 있습니다.

```
CLASSIFIER_URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/2"
IMAGE_RES = 224
model = tf.keras.Sequential([
    hub.KerasLayer(CLASSIFIER_URL, input_shape=(IMAGE_RES, IMAGE_RES, 3))
    layers.Dense(2, activation='softmax')
])

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

EPOCHS = 6

history = model.fit(train_batches,
                    epochs=EPOCHS,
                    validation_data=validation_batches)
```

참고사항

1. 샘 github (접속 후 ai_vision2 클릭)
<https://github.com/sckimclass>
2. 삼성 주니어 AI 에디터
<https://middle.jsa.ai>
3. 강의 참고 자료
<https://www.udacity.com/course/intro-to-tensorflow-for-deep-learning--ud187>
<https://www.tensorflow.org/lite>
<https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>

Tensorflow Lite

1. TensorFlow Lite 인터프리터를 설치

```
$ pip3 install https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-l  
inux_armv7l.whl
```

2. 추론 실행

추론을 실행하려면 다음에서 인터프리터를 가져와야 합니다.

```
from tflite_runtime.interpreter import Interpreter
```

3. 이미지 분류

① 애플리케이션과 모델을 다운로드

```
$ wget https://video.udacity-data.com/topher/2019/September/5d8e6a5f_image-classificat  
ion/image-classification.zip  
$ unzip image-classification.zip  
$ cd image-classification  
$ wget https://storage.googleapis.com/download.tensorflow.org/models/tflite/mobilenet_v  
1_1.0_224_quant_and_labels.zip  
$ unzip mobilenet_v1_1.0_224_quant_and_labels  
$ ls -l  
total 7204  
-rwxr-xr-x 1 pi pi    2396 Aug  6  2019 classify.py  
-rw-r--r-- 1 pi pi   10484 Feb 27  2019 labels_mobilenet_quant_v1_224.txt  
drwxrwxr-x 2 pi pi    4096 Feb 27  2019 __MACOSX  
-rw-r--r-- 1 pi pi 3069250 Feb 27  2019 mobilenet_v1_1.0_224_quant_and_labels.zip  
-rw-r--r-- 1 pi pi 4276352 Aug  3  2018 mobilenet_v1_1.0_224_quant.tflite
```

② 필요한 모듈을 설치

```
$ cat requirements.txt
numpy==1.16.4
Pillow==6.1.0
$ pip3 install -r requirements.txt
```

③ 테스트 이미지를 다운로드

```
$ wget -O input.jpg https://storage.googleapis.com/download.tensorflow.org/example_images/grace_hopper.jpg
```

④ 모델을 실행

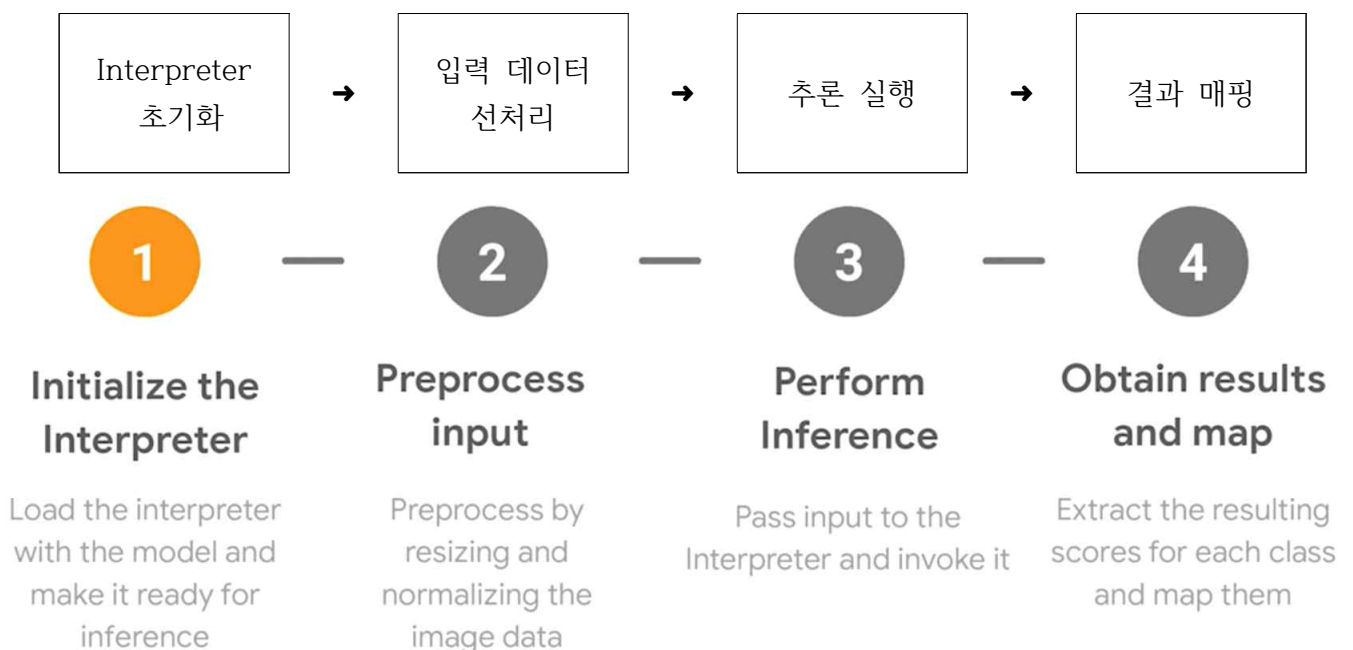
```
$ python3 classify.py --filename input.jpg --model_path mobilenet_v1_1.0_224_quant.tflite --label_path labels_mobilenet_quant_v1_224.txt
```

* 문제 해결

Python 파일을 실행하려고 할 때 libf77blasPython 와 관련된 에러가 발생할 수 있습니다. 오류가 발생하면 다음 명령어를 실행하여 atlas 라이브러리를 설치하십시오.

```
$ sudo apt-get install libatlas-base-dev
...
After this operation, 32.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] <== Y입력하고 Enter키 누르세요.
...
```

4. 이미지 분류 과정



① 인터프리터 초기화

Interpreter를 초기화하고 MobileNet 모델을 로드하겠습니다. 모델을 위한 작업 메모리를 위한 tensor를 할당하겠습니다.

```
# Load the model and allocate tensors
interpreter = Interpreter(model_path=mobilenet_v1_1.0_224_quant.tflite)
interpreter.allocate_tensors()
```

Interpreter로 부터 input tensor와 output tensor 를 알아내고, input tensor에는 입력데이터를 output tensor에는 이미지 분류 결과를 리턴할 것입니다.

```
# Load the model and allocate tensors
interpreter = Interpreter(model_path=mobilenet_v1_1.0_224_quant.tflite)
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
...
```

② 이미지 선처리

이미지 크기를 조절하고 정규화하는 단계입니다. 파일로부터 이미지 파일을 읽고 이미지 크기를 변경하고 정규화하겠습니다. 그다음 모델에서 요구되는 배열(batch)의 차원을 추가하겠습니다.

```
# Read image
img = Image.open(filename).convert('RGB')

# Preprocess image
img = img.resize((244, 244))
input_data = np.array(img)

# Add a batch dimension
input_data = np.expand_dims(img, axis=0)
```

③ 추론 실행

이전에 우리는 추론 실행전 전처리된 이미지를 인터프리터에 입력하고 모델을 실행했습니다.

```
# Point the data to be used for testing and run the interpreter
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
```

④ 결과 매핑

output를 가지고 이미지가 어떤 동물에 속하는지 결정할 수 있습니다. Numpy.argmax 메서드는 결과 중에 가장 가능성이 높은 인덱스를 리턴합니다. Cat and Dog에서는 두 개의 클래스가 있기 때문에 간단히 0이면 Cat, 1이면 Dog라고 예측할 수 있습니다.

```
# Obtain results and print the predicted category
predictions = interpreter.get_tensor(output_details[0]['index'])
predicted_label = np.argmax(predictions)
print('Cat' if predicted_label == 0 else 'Dog')
```

⑤ 높은 순위로 결과 출력하기



dog	0.91
cat	0.07
rabbit	0.02

```
# Obtain results and print the predicted category
predictions = interpreter.get_tensor(output_details[0]['index'])

# Get indices of the top k results
top_k_indices = np.argsort(predictions)[::-1][:top_k_results]

for i in range(top_k_results):
    print(labels[top_k_indices[i]],
          predictions[top_k_indices[i]]/255.0)
```

5. 라즈베리 파이에서 카메라 사용하기

구글에서 raspberry pi camera 검색

<https://projects.raspberrypi.org> > projects > getting-start... ▼

Getting started with the Camera Module - Introduction

Introduction. Learn how to connect the **Raspberry Pi Camera** Module to your Raspberry Pi and take pictures, record video, and apply image effects.

[Take still pictures with Python...](#) · [How to change the image](#) · [Print this project](#)

6. Raspberry Pi에서 이미지 분류

TensorFlow 설치 학습 ▼

For Mobile & IoT

① 개요 튜토리얼 가이드 예시 API

TensorFlow Lite 튜토리얼

비전

- 필기 입력된 숫자 인식
- Android에서 꽃 인식
- iOS에서 꽃 인식
- ② 이미지 부러뜨리기 위한 전이 학습
- Raspberry Pi에서 이미지 분류
- 객체 감지를 위한 전이 학습
- Raspberry Pi에서 객체 감지
- 모바일 객체 감지기 학습

7. 나만의 모델 만들기

TensorFlow 설치 학습 ▼ API ▼ 리소스 ▼ 커뮤니티 ▼ TensorFlow를 사용해야 하는 이유 ▼

For Mobile & IoT ①

개요 튜토리얼 가이드 예시 API

TensorFlow Lite 가이드

시작하기

- Android 빠른 시작
- iOS 빠른 시작
- Python 빠른 시작
- FAQ
- 로드맵

모델 변환

- 개요
- RNN 모델 변환
- ▶ 메타데이터 추가
- 샘플 모델
- API 업데이트

② 모델 만들기

TensorFlow Lite Model Maker

ML 커뮤니티 데이는 11월 9일입니다! TensorFlow, JAX에서 업데이트를 우리와 함께

TensorFlow > 학습 > For Mobile & IoT > 가이드

TensorFlow Lite

TensorFlow Lite는 개발자가 모바일, 내장형 기기, IoT 기기에서 모델을 실행할 수 있도록 하는 도구 모음입니다.

주요 특징

- 기기 내 머신러닝에 최적화됨, 5가지 핵심 제약사항 해결: 지연 시간(서버까지의 왕복 시간을 남기지 않음), 연결성(인터넷 연결이 필요하지 않음), 크기(모델 및 바이너리 크기가 크지 않음)
- 여러 플랫폼 지원: Android 및 iOS 기기, 내장형 Linux 및 마이크로 컨트롤러 등

tflite_model_maker.image_classifier.create

Loads data and retrains the model based on data for image classification.

```
@classmethod
tflite_model_maker.image_classifier.create(
    train_data, model_spec='efficientnet_lite0', validation_data=None,
    batch_size=None, epochs=None, train_whole_model=None, dropout_rate=None,
    learning_rate=None, momentum=None, shuffle=False, use_augmentation=False,
    use_hub_library=True, warmup_steps=None, model_dir=None, do_train=True
)
```

Used in the notebooks

Used in the tutorials

- [Image classification with TensorFlow Lite Model Maker](#)
- [Fine tuning models for plant disease detection](#)

Args

train_data	Training data.
model_spec	Specification for the model.
validation_data	Validation data. If None, skips validation process.
batch_size	Number of samples per training step. If use_hub_library is False, it represents the base learning rate.
epochs	Number of epochs for training.
train_whole_model	If true, the Hub module is trained together with the classification layer on top. Otherwise, only the model is trained.
dropout_rate	The rate for dropout.
learning_rate	Base learning rate when train batch size is 256. Linear to the batch size.
momentum	a Python float forwarded to the optimizer. Only used when use_hub_library is True.
shuffle	Whether the data should be shuffled.
use_augmentation	Use data augmentation for preprocessing.
use_hub_library	Use make_image_classifier_lib from tensorflow hub to retrain the model.
warmup_steps	Number of warmup steps for warmup schedule on learning rate. If None, the default warmup steps are used when use_hub_library is False.
model_dir	The location of the model checkpoint files. Only used when use_hub_library is False.
do_train	Whether to run training.

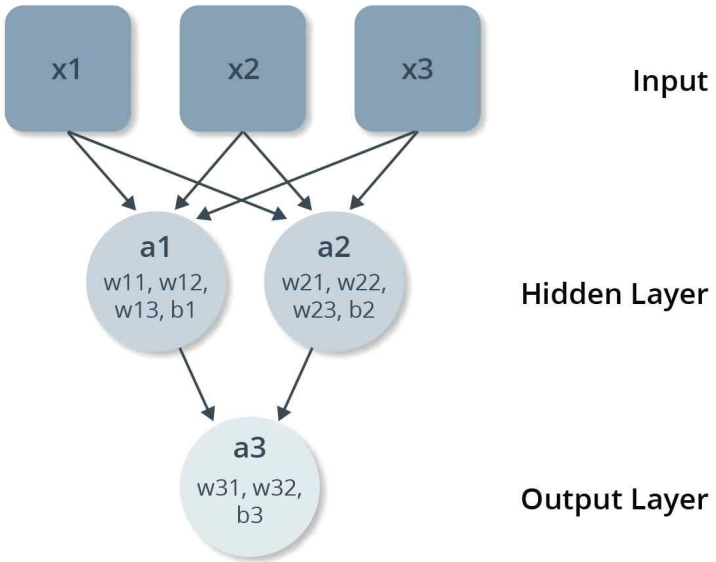
Returns

An instance based on
ImageClassifier.

1. 입력값과 출력값이 다음과 같을때 38을 입력할 때 출력값이 뭐가 될가요?

Input: 0, 8, 15, 22, 38
Output: 32, 46.4, 59, 71.6, ?

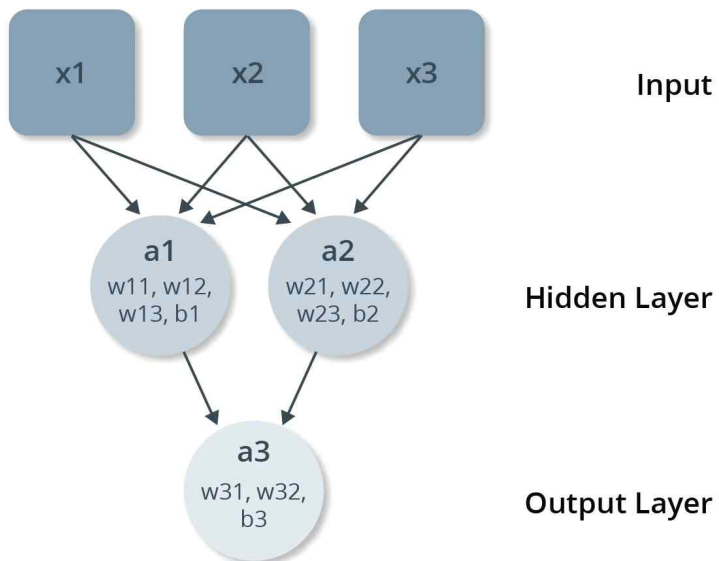
2. 다음은 신경망을 시각화 한 그림이다. a2, a3 뉴런의 변수값을 계산하는 식을 쓰세요.



뉴런	계산 식
a1	$a1 = x1 \cdot w11 + x2 \cdot w12 + x3 \cdot w13 + b1$
a2	
a3	

3. 어떤 인공 신경망의 입력 특성이 100개이고 밀집층(Dense Layer)에 있는 뉴런 개수가 10개일 때 필요한 모델 파라미터의 개수는 몇 개인가요?

4. 다음과 같이 신경망이 구성되어 있을 때 Keras의 Sequential()를 이용하여 모델을 구성하는 Python 코드를 작성하시오. 단, activation 함수는 생략합니다.



5. 다음과 같은 입력에서 (2, 2) 커널을 적용하여 컨볼루션을 적용합니다. 이 컨볼루션 결과를 계산해 보세요.

1	0	1	0
0	1	1	0
0	0	1	1
0	0	1	0

<입력>

1	0
0	1

<커널>

<결과>

6. 다음과 같은 특성 맵이 있습니다. (2, 2) 최대 풀링의 결과를 계산해 보세요.

9	10	6	2
2	1	4	8
9	0	7	1
8	8	3	3

Max Pooling

2 x 2

Stride 2

1	0
0	1

7. 컨볼루션 신경망에서 다음과 같은 필터가 학습되었습니다. 이 필터를 사용해 가장 높은 값의 특성 맵을 만들 수 있는 입력은 무엇일까요?

