<Term Project 2021: Survey on Programming Paradigms>

Deep Learning with Python and comparison between popular frameworks (TensorFlow vs PyTorch)

Table Of Contents

The reason for choosing this topic

img1

img2

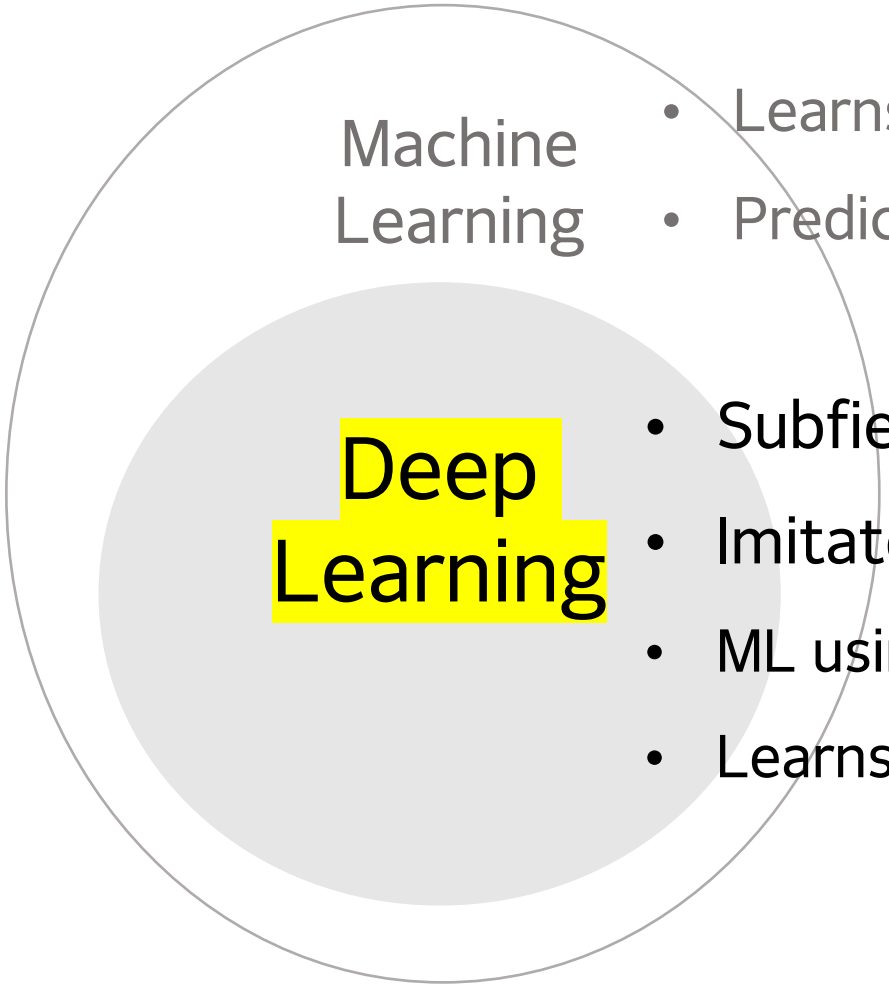
To find out about deep learning,

an area that has recently been gaining much popularity,

and to compare the most in demand frameworks.

Deep Learning overview

- What is deep learning?[1]



- Learns characteristics and patterns from the given data
- Predicts future results for unknown data based on “learning”

- Subfield of ML
- Imitates information-processing system in the human brain
- ML using multiple artificial neural networks consist of neurons
- Learns important characteristics of data, on the machine’s own

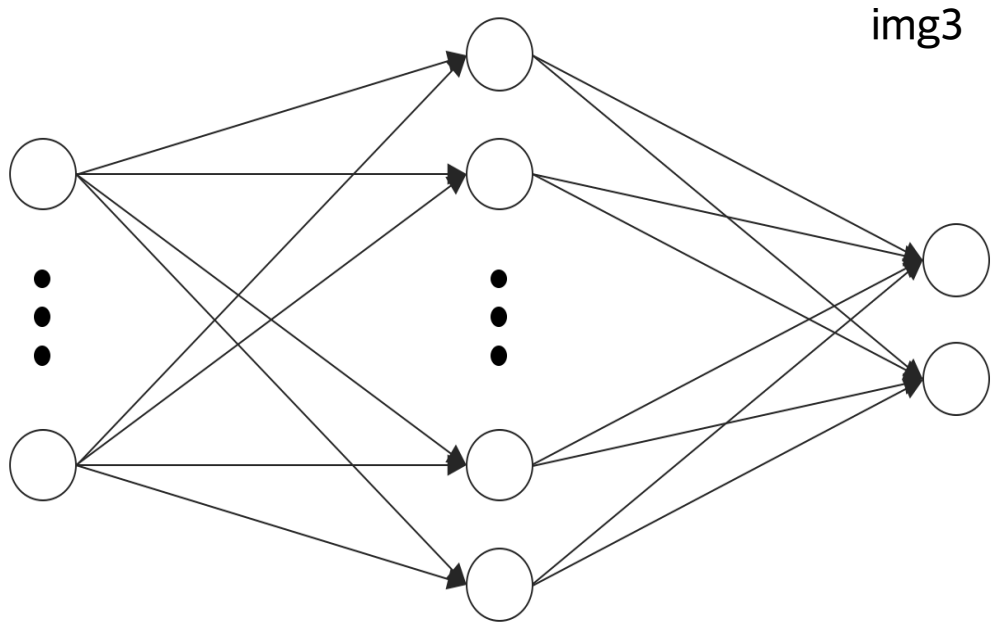
Deep Learning overview

- What is deep learning? (Cont.)[2]

- Composed of over three Hidden layers

→ “Deep” means the model has multiple hidden layers

- Connection between neurons by W and bias
- Updates W and bias at each training epoch to find optimal weights and biases to accurately characterize the data



Deep Learning overview

- Why do we use DL? [3] [4] [5]

- Highly accurate
- Learns faster than humans
- Simple data preprocessing(compared to ML)

  → because the model itself can find the characteristics of the data

- The great improvement of algorithms
- Improved H/W: GPU used to reduce computational time.
- Big data: large amounts of data and tag information available on social media

Deep Learning overview

- Deep learning use cases

img4

Computer Vision

img5

Voice Recognition

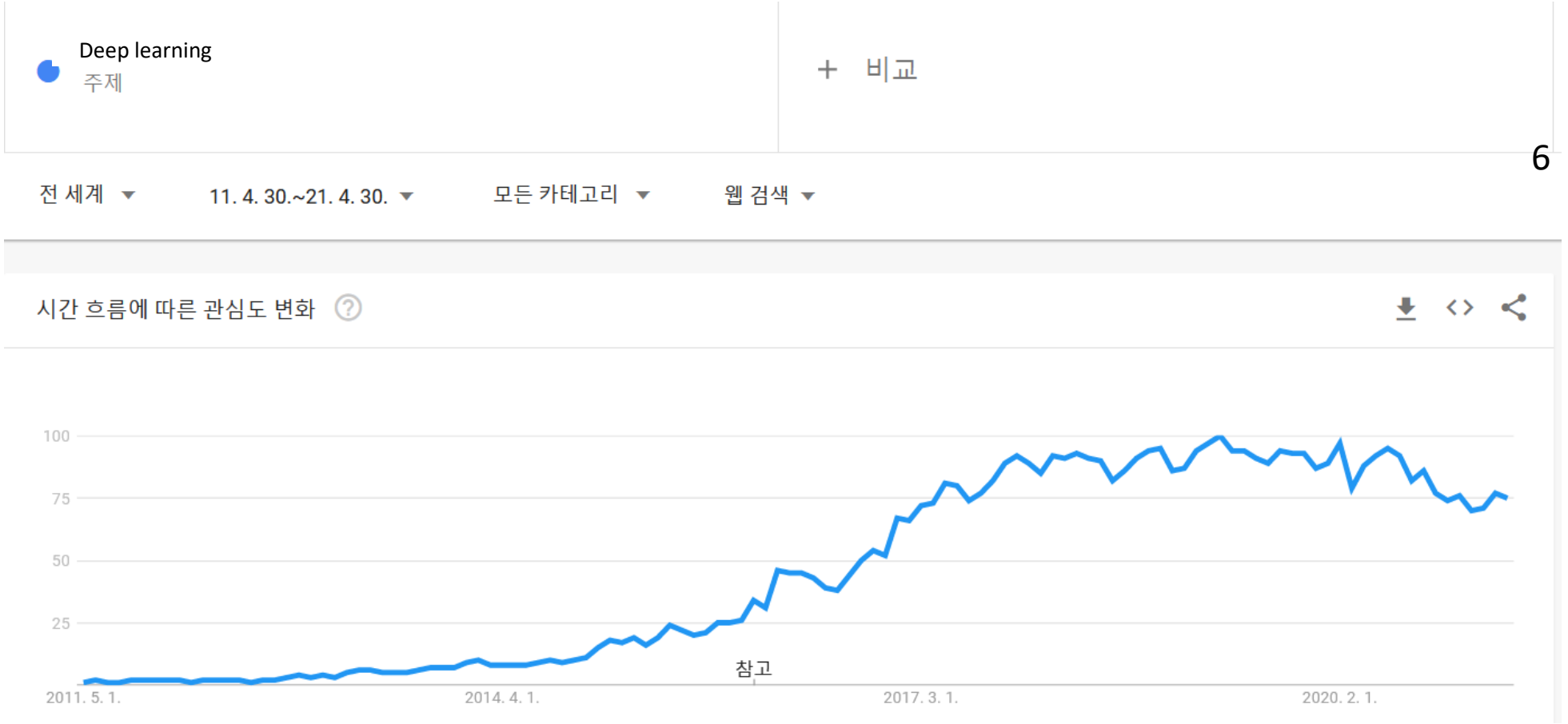
img6

Natural Language Processing

→ Outstanding performance in the above areas;
applications have been developed and widely used

Deep Learning overview

- DL trends →a significant increase in interest compared to 10 years ago



※ Why Python?

img7

- The reasons why Python is primarily used in ML/DL[7]

1. Simple code→ allows to focus on troubleshooting, not on the technical side
2. Various libraries and frameworks → remarkably efficient!
3. Platform independence → executable on a variety of OS
4. Community → allows to have constructive feedback

TensorFlow / PyTorch?

- What is DL framework?[8]
  - Framework provides classes and libraries for application development
  - **DL Framework** offers broad support for ML algorithms

    as well as useful libraries

- While using DL frameworks, you can enjoy…[9]

[Faster learning: programs can be run on GPU

Convenience: designing a model is simple and it eliminates complex calculations]

TensorFlow / PyTorch?

- Examples of DL frameworks

img8

img9

img10

img11

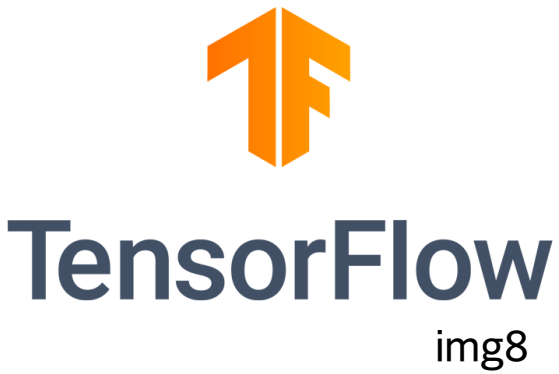
img12

img13

Etc.

TensorFlow / PyTorch?

- TensorFlow overview[10 11]



img8

- Release date: November 2015
- Developer: Google
- Stable release: 2.4.1
- Programming languages: Python, C
- Platforms: Linux, MS Windows, MacOS, Android, iOS, JavaScript
- In short: end-to-end open source platform for ML
- Link: https://www.tensorflow.org/

TensorFlow / PyTorch?

- PyTorch overview[12]

img10

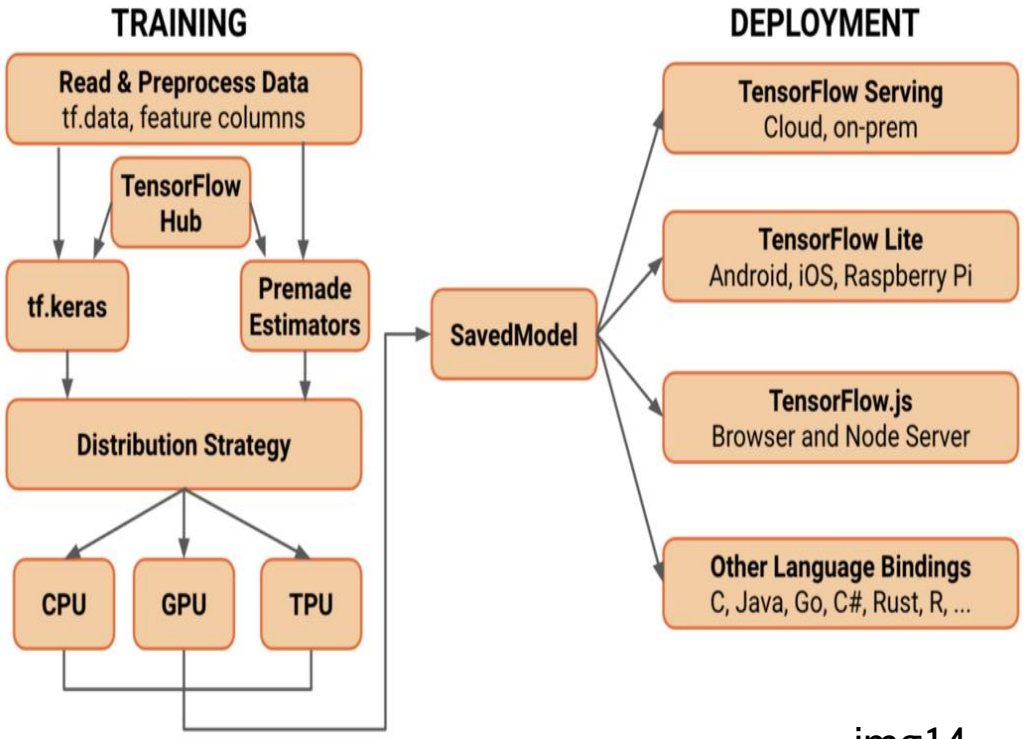
- Release date: October 2016
- Developer: Facebook's AI Research lab (FAIR)
- Stable release: 1.8.0
- Programming languages: Python
- Platforms: Linux, MS Windows, MacOS
- In short: Python-based scientific computation package - used for GPU computation and DL
- Link: https://pytorch.org/

TensorFlow / PyTorch?

- Major features of TensorFlow[13 14 15]

- TensorFlow 2.0(2019)- improved simplicity, convenience (it was less intuitive in TensorFlow 1.0)
- Can be used on a variety of platforms (Mobile, JS)
- Supports from training to deployment
- Easy to use, thanks to Keras API (TensorFlow 2.0)
- Static, or dynamic computational graph can be used (TensorFlow 2.0)
- Model on training can be visualized(using TensorBoard)

TensorFlow 2.0 Diagram



img14

TensorFlow / PyTorch?

- Major features of PyTorch[14] [15]

- "Pytonic" → easy to learn for Python programmers
- Can be debugged with Python debugging tools
- Dynamic computational graph is used
- Well-documented and has an official forum
- Intuitive model creation and convenience were unique advantages of PyTorch,
- but after the release of TensorFlow 2.0, there is no significant difference

15

※ Computational Graph:
Graph of the calculation process using nodes and edges

Static Computational Graph:
Parameters fixed during model training

Dynamic Computational Graph:
Allows to change parameters during model training

→although static graphs are faster, dynamic graphs are more flexible, therefore more preferred

TensorFlow vs PyTorch

- Accuracy/Memory usage/Training time comparison

16

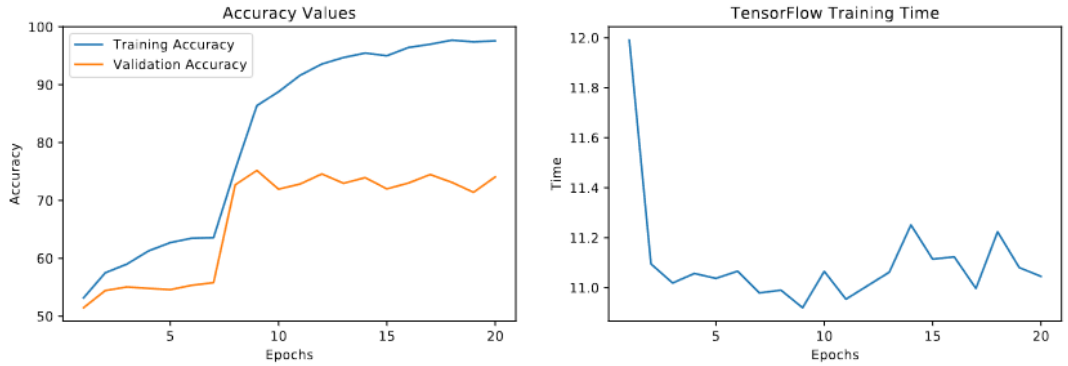


Figure 1: TensorFlow Accuracy and Training Time

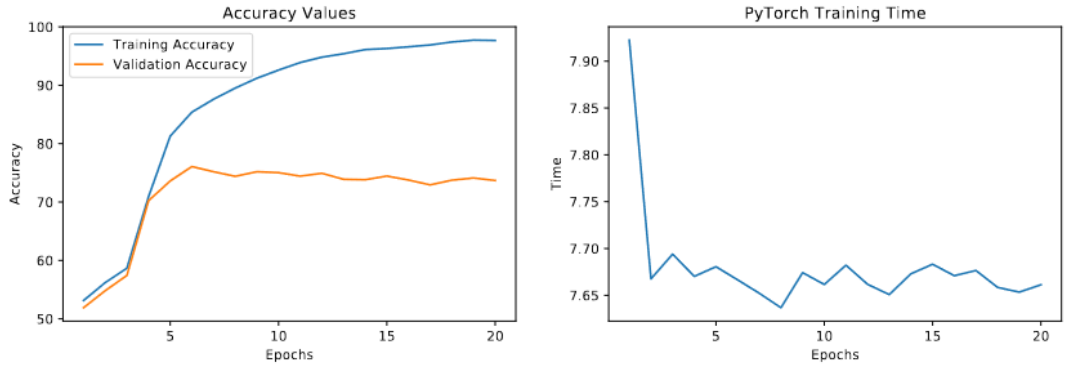


Figure 2: PyTorch Accuracy and Training Time

- Accuracy: no significant difference
- Training time: PyTorch is faster

  (Avg. TensorFlow: 11.19s / Avg. PyTorch: 7.67s)

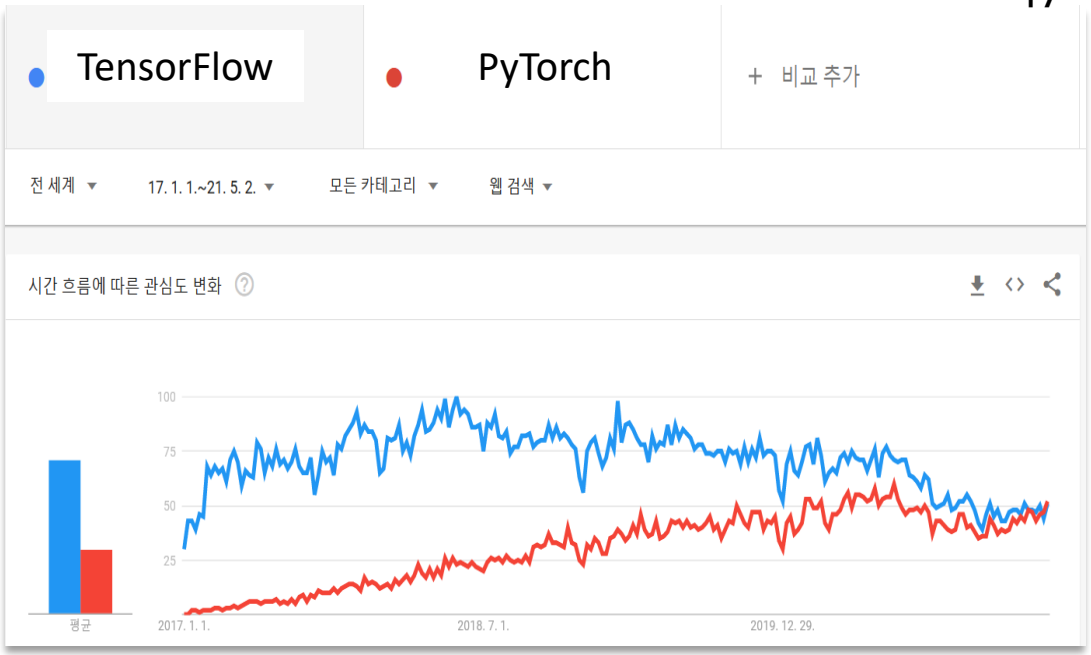
- Memory usage: TensorFlow is lower

  (TensorFlow: RAM 1.7 GB / PyTorch: RAM 3.5 GB)

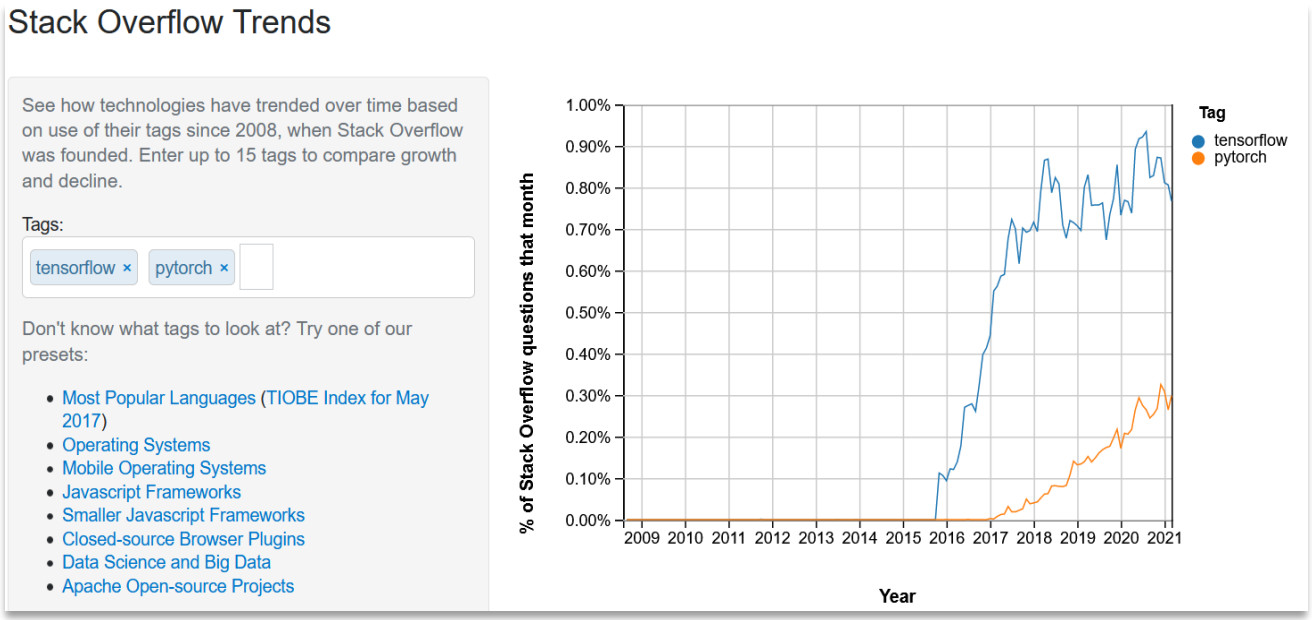
TensorFlow vs PyTorch

- Trend comparison (Google, Stack Overflow)

17



18



→ Stack Overflow trends show overwhelming TensorFlow users

→ However, interest in PyTorch grows rapidly;

PyTorch searches and TensorFlow searches are becoming similar

TensorFlow vs PyTorch

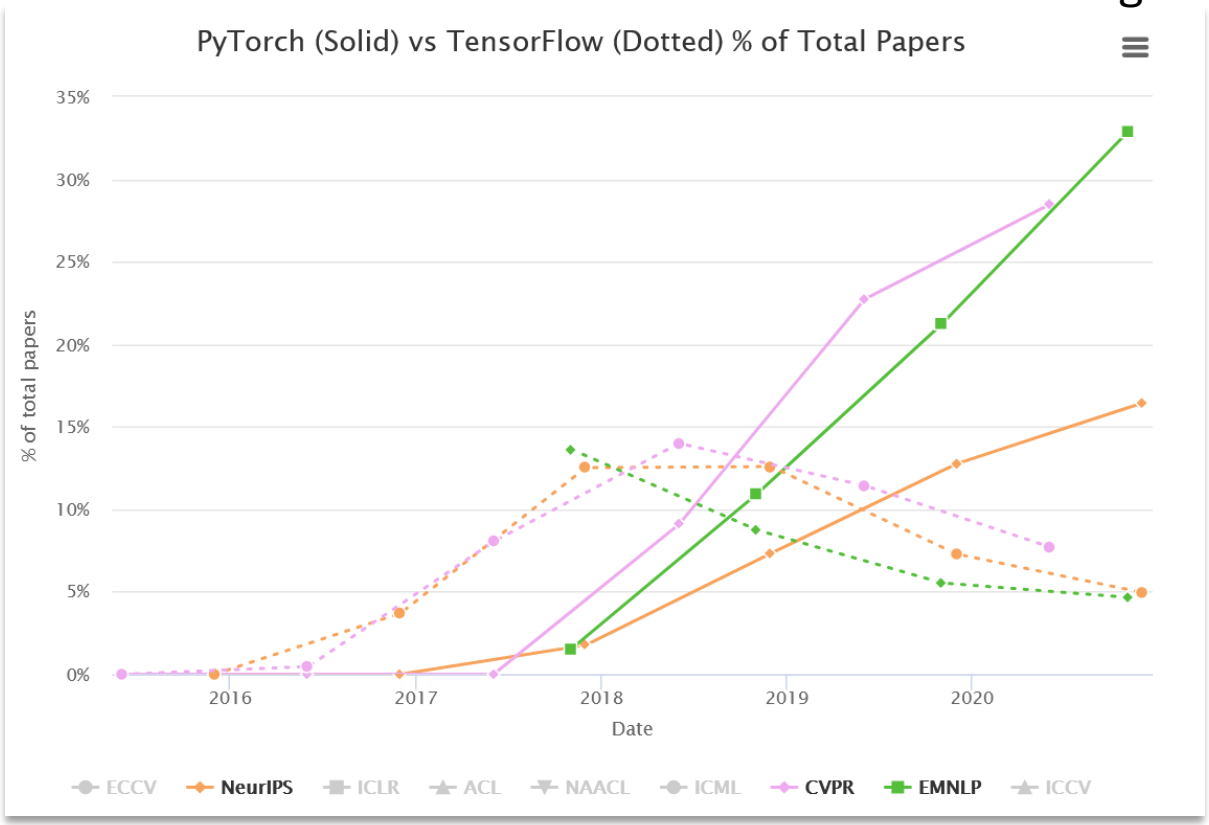
- Trend comparison[19]

State of AI Report 2020 indicates that…

- Paper implementations on GitHub
  → 47% based on PyTorch (18% TensorFlow)

- 20-35% of conference papers mentioned the framework they use
  → 75% cite the use of PyTorch

img15



- Authors who published more TensorFlow papers than PyTorch papers in 2018 55% of them have switched to PyTorch (15% for the opposite case)

TensorFlow vs PyTorch

- Environment: Google Colaboratory
- Purpose: to build a simple CNN model to compare code and accuracy
- Dataset to use: MNIST

(full code: https://github.com/bobaejeon/pl2021)

img16

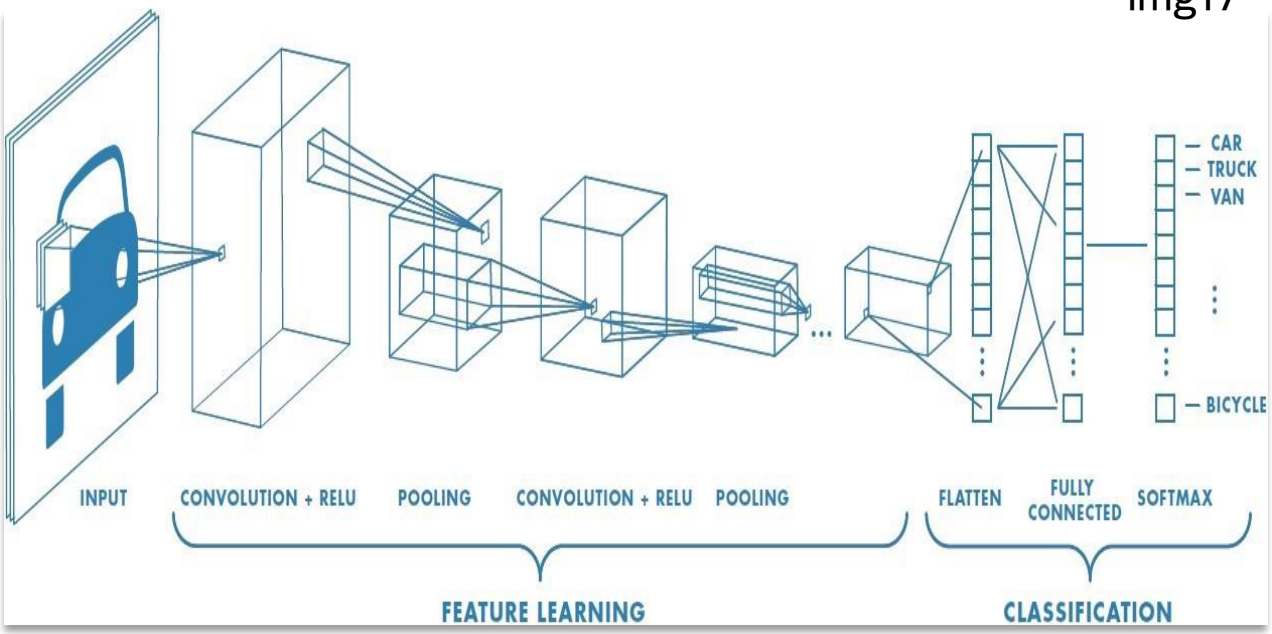
- Handwriting digits(from 0 to 9) database[20]
- 60,000 images to train, 10,000 images to test

  → 28x28 each
- "Hello world" of ML

※ What is CNN?

- Convolutional Neural Network[21]

img17



- Widely used in computer vision
- Convolution Layer, Pooling Layer

[Convolution Layer: finds out vital features

Pooling Layer: downsizes the sample

→prevents overfitting / allows faster computation

(overfitting: an analysis corresponds too closely to a particular dataset, therefore, may fail to predict different dataset)

TensorFlow vs PyTorch: code comparison

- import and load data: TensorFlow

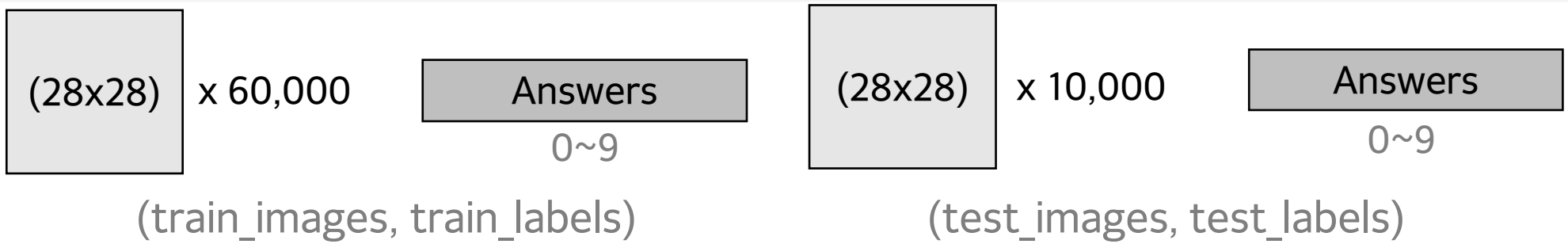
- MNIST is extremely popular, so it can be easily downloaded by using built-in functions in each framework.

```
import tensorflow as tf
import matplotlib.pyplot as plt ←To show sample images
```

Keras library will be used

```
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

print(train_images.shape, train_labels.shape, test_images.shape, test_labels.shape)
```

```
(60000, 28, 28) (60000,) (10000, 28, 28) (10000,)
```



TensorFlow vs PyTorch: code comparison

- import and load data: PyTorch

```
import torch
import torchvision
import matplotlib.pyplot as plt


train_dataset = torchvision.datasets.MNIST(root='./data/',
                                           train=True,
                                           transform=torchvision.transforms.ToTensor(),
                                           download=True)
test_dataset = torchvision.datasets.MNIST(root='./data/',
                                          train=False,
                                          transform=torchvision.transforms.ToTensor(),
                                          download=True)

train_dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=32)
test_dataloader = torch.utils.data.DataLoader(test_dataset, batch_size=32)
```

Training data? True : False

Has to be converted into tensor data type in order to compute

→ Like iterator

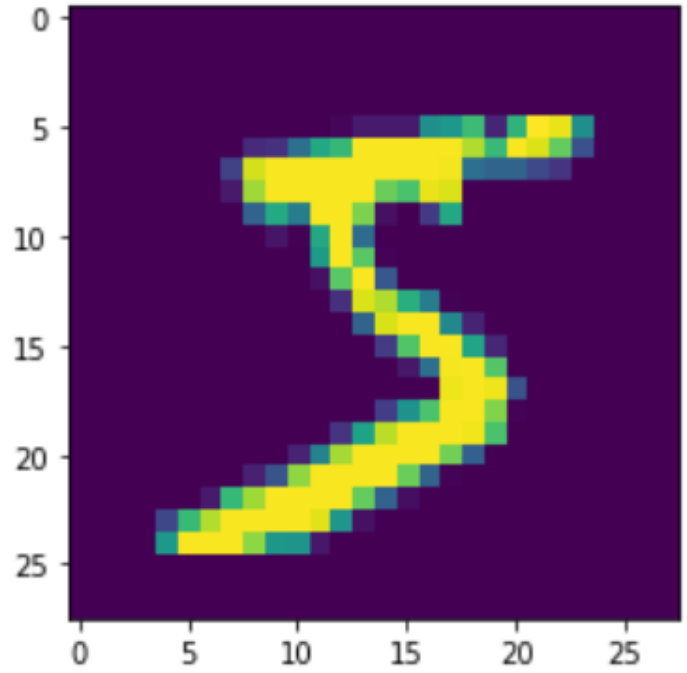
※ Tensor: Array of data (over 3D)

TensorFlow vs PyTorch: code comparison

- Show sample images

```
plt.imshow(train_images[0])
print(train_labels[0])
```

5

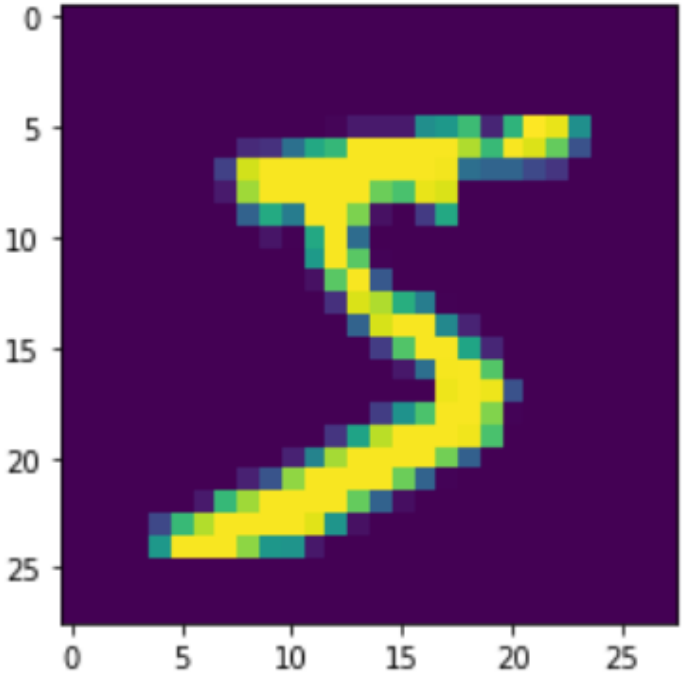


TensorFlow

```
image, label = train_dataset[0]

plt.imshow(image.squeeze())
print(label)
```

→ Eliminates dimensions if they're 1

5



PyTorch

TensorFlow vs PyTorch: code comparison

- CNN Model: TensorFlow

- Data preprocessing: reshape and normalize

Input: tensor
(number of imgs, height, width, color channels)

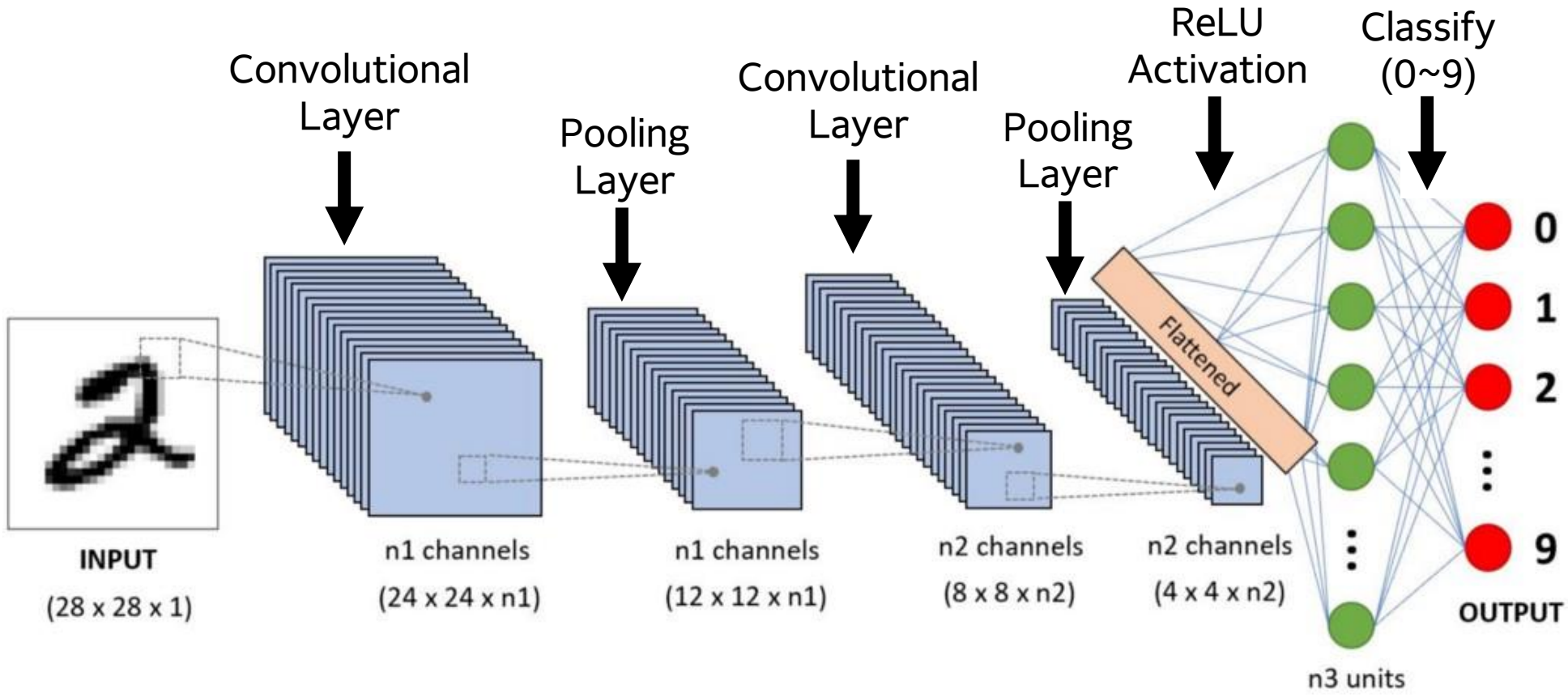
```
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# normalize
train_images, test_images = train_images / 255.0, test_images / 255.0
```

↑ Normalize(0~1)

TensorFlow vs PyTorch: code comparison

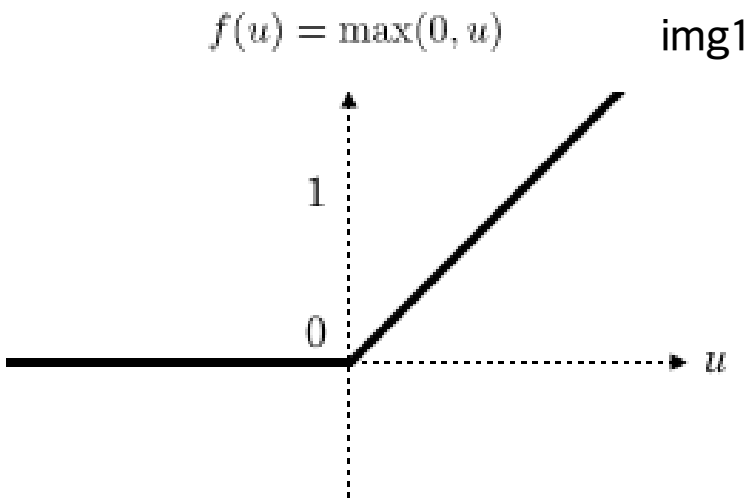
- Model structure to be used

img18



※ Activation function[22]

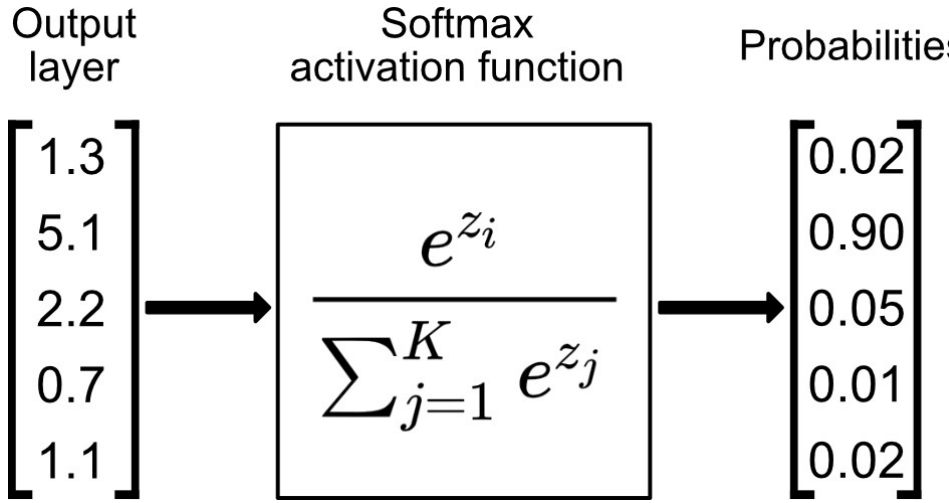
- Defines the output of that node given an input or set of inputs
- Makes each layer a nonlinear combination
- If not used, the output for the input will appear linear
- There are plenty of it, however, we will be using ReLU and Softmax



img19

ReLU(Rectified Linear Unit)
Keeps positive values only

img20



Softmax
Output: probability for each class

TensorFlow vs PyTorch: code comparison

- CNN model: TensorFlow

- Using Keras Sequential API

# of output filters

Uses filter sized (5x5)

Shape of each sample: (28, 28, 1)

```
modeltf = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(5, 5), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(5, 5), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=64, activation='relu'),
    tf.keras.layers.Dense(units=10, activation='softmax')
])
```

→ Convert 3D array into 1D

→ Classify the images into 10 classes

Will take the maximum value
in an area sized (2x2)

TensorFlow vs PyTorch: code comparison

- CNN model: PyTorch

- Inherits nn.Module, initializer and forward method must be implemented

```
class PyNet(torch.nn.Module):
  def __init__(self):
        super().__init__()
        self.flatten = torch.nn.Flatten()
        self.cnn_model = torch.nn.Sequential(
            torch.nn.Conv2d(in_channels=1, out_channels=32, kernel_size=(5, 5)),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=(2, 2)),
            torch.nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(5, 5)),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=(2, 2)),
            torch.nn.ReLU()
        )
        self.fc_model = torch.nn.Sequential(
            torch.nn.Linear(in_features=4*4*64, out_features=64),
            torch.nn.ReLU()
        )
        self.classifier = torch.nn.Linear(64, 10)

  def forward(self, x):
        x = self.cnn_model(x)
        x = self.flatten(x)
        x =  self.fc_model(x)
        out = self.classifier(x)
        return out
```

Same model as the one using TF

forward: defines “how” to work

※ Loss function and Optimizer

- Loss function[23]
- Evaluates how accurately the model predicts data
- Calculates the error between the predicted value and the actual value
- The lower, the better

- Optimizer[24]
- Allows faster and stable training
- Adjusts weights and bias in the direction of reducing loss

TensorFlow vs PyTorch: code comparison

- Loss function and Optimizer

- Loss function used: Categorical Cross Entropy Loss
- Optimizer used: Adam Optimizer

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

TensorFlow

```
loss_fn = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(modelPy.parameters())
```

PyTorch

TensorFlow vs PyTorch: code comparison

- Train the model: TensorFlow
- Using fit() method, train for 5 times

```
modeltf.fit(train_images, train_labels, epochs=5)
# 60000 examples / 32 batch size = 1875 number of batches
```

Loss decreases and accuracy increases

```
Epoch 1/5
1875/1875 [==============================] - 55s 29ms/step - loss: 0.2914 - accuracy: 0.9097
Epoch 2/5
1875/1875 [==============================] - 54s 29ms/step - loss: 0.0400 - accuracy: 0.9884
Epoch 3/5
1875/1875 [==============================] - 53s 28ms/step - loss: 0.0262 - accuracy: 0.9916
Epoch 4/5
1875/1875 [==============================] - 53s 28ms/step - loss: 0.0211 - accuracy: 0.9935
Epoch 5/5
1875/1875 [==============================] - 53s 28ms/step - loss: 0.0157 - accuracy: 0.9946
```

TensorFlow vs PyTorch: code comparison

- Train the model: PyTorch
- All the steps must be stated unlike TF

```
def train(dataloader, model, loss_fn, optimizer, epoch):
    model.train() ← Set: train mode
    size = len(dataloader.dataset)
    for e in range(epoch):
      loss, test_loss, correct = 0, 0, 0
      for X, y in dataloader:
        # Compute prediction and loss
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        test_loss = loss.item()
        correct += (pred.argmax(1) == y).type(torch.float).sum().item()

      print(f"Epoch: {e+1}, loss: {test_loss:>5f}, accuracy: {(correct / size):>5f}")
```

Result:

```
Epoch: 1, loss: 0.028312, accuracy: 0.954533
Epoch: 2, loss: 0.003180, accuracy: 0.985700
Epoch: 3, loss: 0.000613, accuracy: 0.990600
Epoch: 4, loss: 0.000633, accuracy: 0.993500
Epoch: 5, loss: 0.000072, accuracy: 0.994700
```

```
train(train_dataloader, modelpy, loss_fn, optimizer, epoch=5) ← Trains for 5 times as well
```

TensorFlow vs PyTorch: code comparison

- Test

```
test_loss, test_acc = modeltf.evaluate(test_images,  test_labels)
# 10000 test imgs / 32 batch size = 313 number of batches
```

```
313/313 [==============================] - 3s 9ms/step - loss: 0.0301 - accuracy: 0.9914
```

TensorFlow

Using evaluate method

```
modelpy.eval()
```

←set: evaluation mode

```
size = len(test_dataloader.dataset)
test_loss, correct = 0, 0

with torch.no_grad():
  for X, y in test_dataloader:
    pred = modelpy(X)
    correct += (pred.argmax(1) == y).type(torch.float).sum().item()

print(f"Test Result- accuracy: {(correct / size):>5f}")
```

Predicts the result and calculates accuracy

```
Test Result- accuracy: 0.989500
```

PyTorch

TensorFlow vs PyTorch: code comparison

- Accuracy

```
test_loss, test_acc = modeltf.evaluate(test_images,  test_labels)
# 10000 test imgs / 32 batch size = 313 number of batches
```

```
313/313 [==============================] - 3s 9ms/step - loss: 0.0301 - accuracy: 0.9914
```

```
modelpy.eval()
size = len(test_dataloader.dataset)
test_loss, correct = 0, 0

with torch.no_grad():
  for X, y in test_dataloader:
    pred = modelpy(X)
    correct += (pred.argmax(1) == y).type(torch.float).sum().item()

print(f"Test Result- accuracy: {(correct / size):>5f}")
```

```
Test Result- accuracy: 0.989500
```

TensorFlow
99.14%

>

PyTorch
98.95%

(0.19% difference)

Conclusion

- Summary and evaluation

- The code itself is very similar (Tensorflow 2.0)
- Uses the same language Python, making little difference in terms of readability, writability, reliability, cost
- TF was more accurate based on the code written earlier, however, there were no significant difference

- **TensorFlow**

  Simple, if using Keras API / besides, it provides different levels of APIs.

  Can be used from building to deploying models in a variety of platforms (such as mobile)

  Memory usage is relatively low when training

- **PyTorch**

  Looks a bit more complicated than code written in TF, but not difficult.

  Easy to debug, relatively less training time.

Conclusion

- So, which one? (personal findings)

- Each framework has its strengths and weaknesses, so it looks like it can be chosen based on the purpose of use.

1. **Looking for a framework has an active community? TensorFlow and PyTorch**

   →However, TF 2.0 appeared only 2 years ago and is considerably different from 1.0,
   which can be confusing when getting information from the Internet.

2. **New to DL and want to use it easily? TensorFlow**

   →Personally, Keras API in TF was relatively simple based on my experience of writing and comparing the code

3. **Want to learn DL seriously? PyTorch**

   →It's a rapidly evolving field and there's a lot to refer to due to the growing trend of papers using PyTorch

4. **Want to use it on embedded systems, on mobile, or deploy applications? TensorFlow**

   →TF reliably supports

References: texts

-Deep Learning overview-

1. (Korean)Book: 머신 러닝을 위한 파이썬 한 조각/박성호/비제이퍼블릭/2020
2. (Korean)알파고 바둑 실력의 비밀, '딥 러닝(Deep Learning) http://news.unist.ac.kr/kor/column_202/
3. Deep Learning https://www.ibm.com/cloud/learn/deep-learning
4. (Korean)딥러닝(위키백과)- 왜 다시 딥러닝인가? https://ko.wikipedia.org/wiki/%EB%94%A5_%EB%9F%AC%EB%8B%9D#%EC%99%9C_%EB%8B%A4%EC%8B%9C_%EB%94%A5_%EB%9F%AC%EB%8B%9D%EC%9D%B8%EA%B0%80?
5. (Korean)1.3 왜 딥러닝일까? https://tensorflow.blog/%EC%BC%80%EB%9D%BC%EC%8A%A4-%EB%94%A5%EB%9F%AC%EB%8B%9D/1-3-%EC%99%9C-%EB%94%A5%EB%9F%AC%EB%8B%9D%EC%9D%BC%EA%B9%8C-%EC%99%9C-%EC%A7%80%EA%B8%88%EC%9D%BC%EA%B9%8C/
6. Google trends(deeplearning) https://trends.google.com/trends/explore?geo=US&q=%2Fm%2F0h1fn8h
7. Why Use Python for AI and Machine Learning? https://steelkiwi.com/blog/python-for-ai-and-machine-learning/

-Tensorflow / PyTorch-

8. Framework definition https://techterms.com/definition/framework
9. Deep Learning Frameworks https://developer.nvidia.com/deep-learning-frameworks
10. TensorFlow guide https://www.tensorflow.org/guide
11. (Korean)텐서플로(위키백과) https://ko.wikipedia.org/wiki/%ED%85%90%EC%84%9C%ED%94%8C%EB%A1%9C
12. PyTorch(README.md) https://github.com/pytorch/pytorch

References: texts

13. PyTorch vs TensorFlow: Difference you need to know https://hackr.io/blog/pytorch-vs-tensorflow
14. Pytorch vs Tensorflow: A Head-to-Head Comparison https://viso.ai/deep-learning/pytorch-vs-tensorflow/
15. TensorFlow vs PyTorch – A Detailed Comparison https://www.machinelearningplus.com/deep-learning/tensorflow1-vs-tensorflow2-vs-pytorch/

-TensorFlow vs PyTorch-

16. A Comparison of Two Popular Machine Learning Frameworks http://www.ccsc.org/publications/journals/SE2019.pdf#page=20
17. Google trends(TensorFlow vs PyTorch) https://trends.google.com/trends/explore?geo=US&q=%2Fg%2F11bwp1s2k3,%2Fg%2F11gd3905v1
18. Stackoverflow trends(TensorFlow vs PyTorch) https://insights.stackoverflow.com/trends?tags=tensorflow%2Cpytorch
19. State of AI Report 2020 https://docs.google.com/presentation/d/1ZUimafgXCBSLsgbacd6-a-dqO7yLyzll1ZJbiCBUUT4/edit#slide=id.g8b560ae0a6_0_49
20. MNIST database https://en.wikipedia.org/wiki/MNIST_database
21. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
22. Activation function https://en.wikipedia.org/wiki/Activation_function
23. Introduction to loss functions https://algorithmia.com/blog/introduction-to-loss-functions
24. Various Optimization Algorithms For Training Neural Network https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

References: images

-The reason for choosing this topic

img1 https://commons.wikimedia.org/wiki/File:AlphaGo.svg

img2 https://www.analyticssteps.com/blogs/deep-learning-overview-practical-examples-popular-algorithms

-Deep learning overview-

img3 http://news.unist.ac.kr/kor/column_202/

img4 https://soft-cluster.com/computer-vision/

img5 https://tweakreviews.com/gadgets/speech-recognition-in-outlook

img6 https://www.slashgear.com/google-translate-finally-gets-new-languages-in-latest-update-26611357/

img7 https://www.python.org/community/logos/

-TensorFlow / PyTorch?-

img8 https://github.com/tensorflow/tensorflow

img9 https://keras.io/

img10 https://icon-icons.com/icon/pytorch-logo/169823

References: images

img11 https://github.com/apache/incubator-mxnet

img12 https://en.wikipedia.org/wiki/Microsoft_Cognitive_Toolkit

img13 https://software.intel.com/content/www/us/en/develop/tools/frameworks.html

img14 https://medium.com/tensorflow/whats-coming-in-tensorflow-2-0-d3663832e9b8

-TensorFlow vs PyTorch-

img15 http://horace.io/pytorch-vs-tensorflow/

img16 https://en.wikipedia.org/wiki/MNIST_database

img17 https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

img18 https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (일부 수정)

img19 https://www.researchgate.net/figure/ReLU-activation-function_fig3_319235847

img20 https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60