PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Machine Learning
Deep Learning

GyeongYeong, Kim
I-Eon, Na

ML/DL Goal

PIAI Research Department

- Keras / Pytorch에 대한 전반적인 Framework에 이해한다.
- 이미지 데이터셋에 대한 분류를 딥러닝 네트워크로 학습할 수 있다.
- 자신이 학습한 모델에 대해 분석할 수 있다.
- 기존의 다른 모델을 가져와 적용할 수 있다.

ML/DL Lab

PIAI Research Department

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- 1일차 : Framework / Classifier(~DNN)
- 2일차 : Analysis Tools / Feature Extraction(~CNN)
- 3일차 : Additional Func.(Data Augmentation / Fine Tuning)
  Customization
- 4일차 : Additional Func.(Data Augmentation / Fine Tuning)
  RNN / GAN

ML/DL Lab 1st Day

PIAI Research Department

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Keras Basic
  - Framework Introduction
  - Y=3X Regression
  - ANN/DNN with MNIST
  - Model 저장,로드
- Pytorch Basic
  - Framework Introduction
  - Y=3X Regression
  - ANN/DNN with MNIST
  - Model 저장,로드

ML/DL Lab 2nd Day

PIAI Research Department

가치창출대학 POSTECH POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Keras Model Analysis
  - Callback
  - History 확인
  - 틀린샘플 확인
  - Confusion Matrix
- Feature Extraction
  - CNN / Pooling Layer
  - Batch Normalization / DropOut
- Pytorch Feature Extraction
  - CNN / Pooling Layer
  - Batch Normalization / DropOut
  - Best Model Saving
  - History 확인
  - Confusion Matrix
  - 틀린 샘플 확인
  - Early Stopping

ML/DL Lab 3rd Day

PIAI Research Department

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Keras Additional Functions
  - CIFAR10 Introduction
  - Model Summation
  - Data Augmentation
  - Transfer Learning
- Pytorch Additional Functions
  - Data Augmentation
  - Transfer Learning
  - Training Customized Dataset(Man/Monkey)

ML/DL Lab 4th Day

PIAI Research Department

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Keras Additional Functions
  - LSTM (with MNIST)
  - GAN (with MNIST)
- Pytorch Additional Functions
  - GAN (with MNIST)
  - Quiz

ML/DL Lab Recommendation

PIAI Research Department

- 인트로 – 코딩 – 직접 실습
- 이론과 다소 별개 : 구체적인 용어 특성이나 성질들은 이론 수업이나 검색으로 해결할 것
- Coding Time : 수업 시간 내 잘 집중해서 진행할 것(수업 이후 정답지 제공)
- 동작 관련 오류 : 각 반 보조에게 손들고 해결 요청
- 숙제 : 혼자 힘으로 다양하게 시도해보면서 배워볼 것
- 퀴즈 : 실습 수업 내용 잘 들으면서 공부할 것

ML/DL Homework

PIAI Research Department

- MNIST 데이터 셋의 학습 / 평가 / 분석 (Keras / Pytorch 중 택1)
- 제출 파일 : 해당 파일이 들어있는 zip 파일 : ex) HW_김경영.zip
  - Homework_(이름).ipynb (모두 Run한 상태로)
  - 보고서_(이름).docx

- 정확도(60%) - 진행 전략(20%) - 분석(20%)을 통합적으로 평가
  - 정확도 : 99.75% 넘으면 만점 (넘으면 굳이 더 학습해보려고 하진 않아도 됨)
  - 진행 전략 : Keras/Torch 라이브러리 내 적용할 수 있는 수준의 네트워크로 학습 (나름 시도해본 Trial들을 간단하게 정리해도 좋음!)
  - 학습 분석 : Under/Over fitting 여부 확인 / 학습 안정화 여부 확인
  - 성능 분석 : 잘못된 분류 케이스 확인 / 개선할 수 있는 방향 제안 / 현실적인 최대 정확도 제안

- 제출 시 최저점수가 40점이므로 늦더라도 제출 할 것 (미제출 : 0점)
- 표절 심증이 있을 시, 최저점수 반영

ML/DL Lab Quiz

PIAI Research Department

- True/False : 6문제 (30)
- 단답형 : 10문제 (30)
- 실습 코드 : 2문제 (20)
- 서술형 : 1문제 (20)

PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

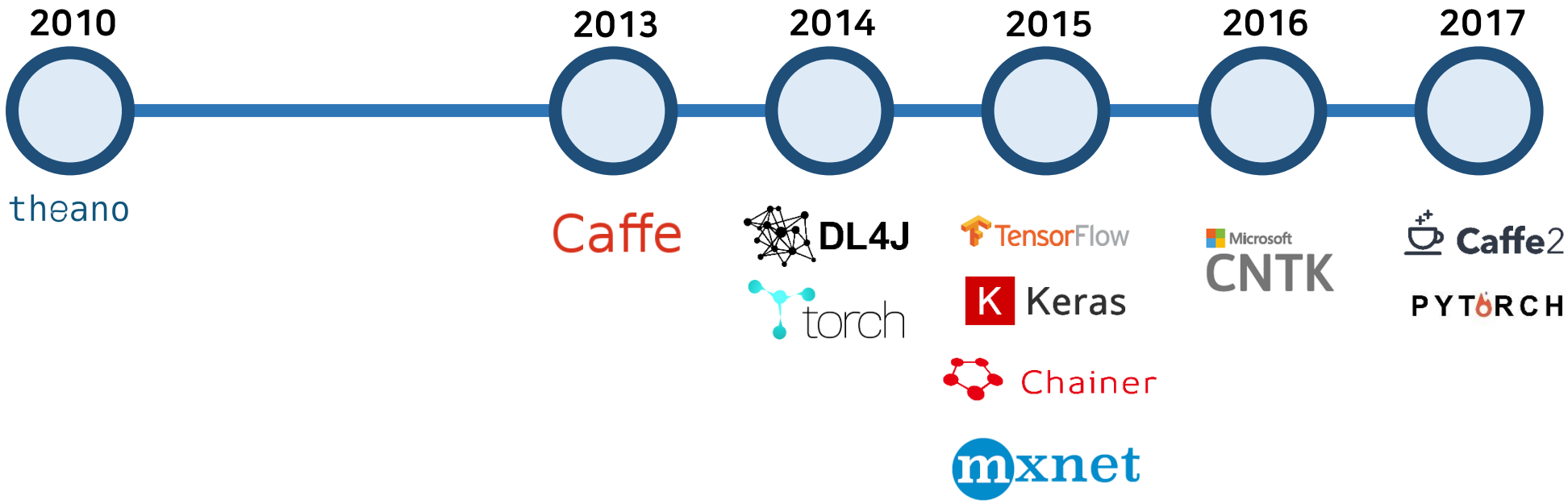
가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Keras Basic

GyeongYeong, Kim

딥러닝 프레임워크 Timeline

PIAI Research Department



딥러닝 프레임워크 비교

PIAI Research Department

주체	플랫폼	모바일	언어	인터페이스	OpenMP	Cuda	OPenCL	멀티GPU	분산
SkyMind	Cross – platform	Android	Java	Java, Scala, Python	Y	Y	-	Y	Y (Spark)

장점

- 가장 많은 프로그래머를 보유하는 Java 기반 딥러닝 프레임워크
- 쉬운 이식성, 높은 안정성
- 문서화가 잘되어 있음
- 시각화 도구 제공

단점

- Java의 언어적 특성으로 인해 학습 및 테스트 과정이 번거로움
- 협소한 사용자 커뮤니티
- 부족한 예제

딥러닝 프레임워크 비교

PIAI Research Department

Caffe2

주체	플랫폼	모바일	언어	인터페이스	OpenMP	Cuda	OPenCL	멀티GPU	분산
BAIR, facebook	Linux, Mac	-	C++	Python, C++	Y	Y	-	Y	-

장점

- 이미지 처리에 특화
- Caffe Model Zoo를 통한 다양한 Pre-trained Model 제공

단점

- 이미지 이외의 데이터 처리에는 부적합(텍스트, 사운드 등)
- 유연하지 못한 API(C++/CUDA로 직접 구현 필요)
- 문서화가 잘 안되어 있음

딥러닝 프레임워크 비교

PIAI Research Department

PYTORCH

주체	플랫폼	모바일	언어	인터페이스	OpenMP	Cuda	OPenCL	멀티GPU	분산
Facebook	Linux, Mac, Window	Android, iOS	Lua	python	Y	Y	Y	Y	Y

장점

- 알고리즘 모듈화가 잘되어 있어 사용이 용이
- 다양한 데이터 전처리 및 시각화 유틸리티 제공
- 직관적인 API
- 다양한 Pre-trained Model 제공

단점

- Lua 언어 자체가 Java/Python보다 폐쇄적
- 상용 어플리케이션이 아니라 연구용으로 적합

딥러닝 프레임워크 비교

PIAI Research Department

theano

주체	플랫폼	모바일	언어	인터페이스	OpenMP	Cuda	OPenCL	멀티GPU	분산
Google	Linux, Mac, Window	-	Python	Python	Y	Y	-	Y	-

장점

- Low-level을 제어할 수 있는 API
- 빠르고 유연함
- Wrapper 프레임워크의 기반 프레임워크(Keras)

단점

- 매우 복잡함
- 2017년 이후 공식 지원 마감

딥러닝 프레임워크 비교

PIAI Research Department

주체	플랫폼	모바일	언어	인터페이스	OpenMP	Cuda	OPenCL	멀티GPU	분산
Google	Linux, Mac, Window	Android, iOS	Python, C++	Python, C/C++, Java, Go	N	Y	-	Y	Y

장점

- Low-level / High level API 모두 제공
- 방대한 사용자 커뮤니티
- 문서화가 잘되어 있음
- 시각화 도구 제공(TensorBoard)

단점

- 런타임 중 그래프 변경 안됨
- 비교적 느림
- CUDA 연결 이슈에 민감함

딥러닝 프레임워크 비교

PIAI Research Department

Keras

주체	플랫폼	모바일	언어	인터페이스	OpenMP	Cuda	OPenCL	멀티GPU	분산
Francois Chollet	Linux, Mac, Window	-	Python	Python	Y(Theano) N(TF)	Y	-	Y	-

장점

- 직관적인 API 인터페이스
- Caffe, Torch, Tensorflow 등 다양한 딥러닝 프레임워크 모델 import 기능 제공
- 문서화가 잘되어 있음

단점

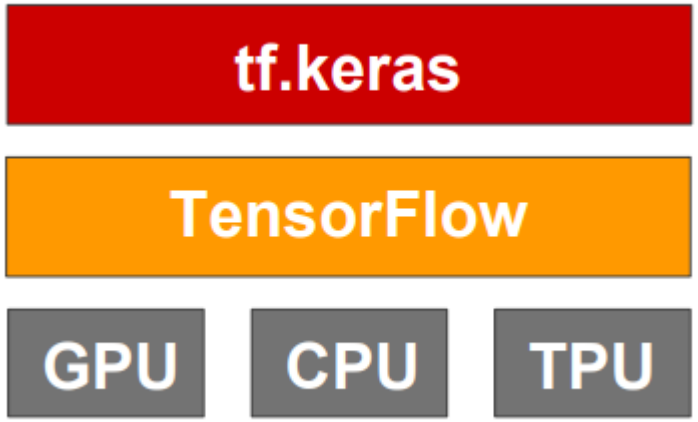
- 텐서플로우의 단점과 유사함

Keras

- tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.14

- In-browser with GPU acceleration (WebKeras, Keras.js, WebDNN…)
- Android (TF, TF Lite), iPhone (native CoreML support), Raspberry Pi, JVm

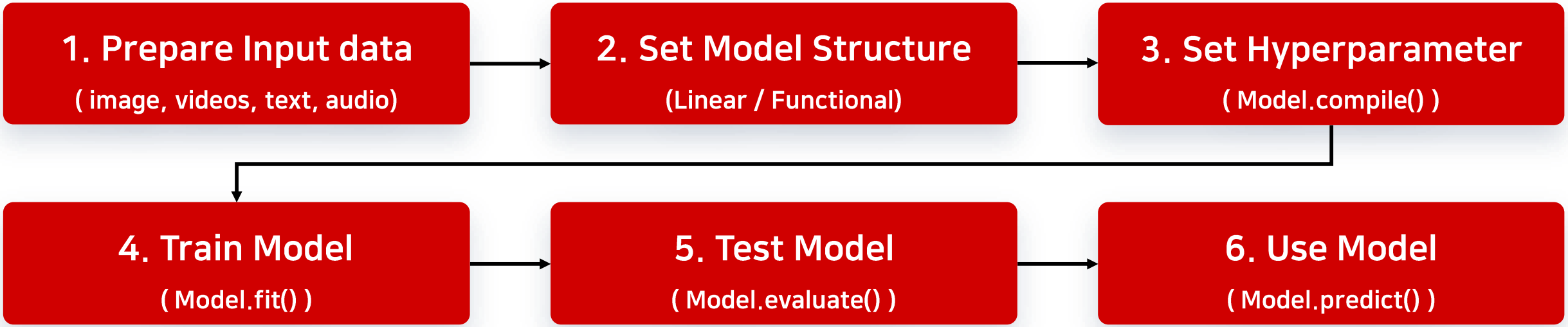
- 쉬움!



Keras Framework Flow

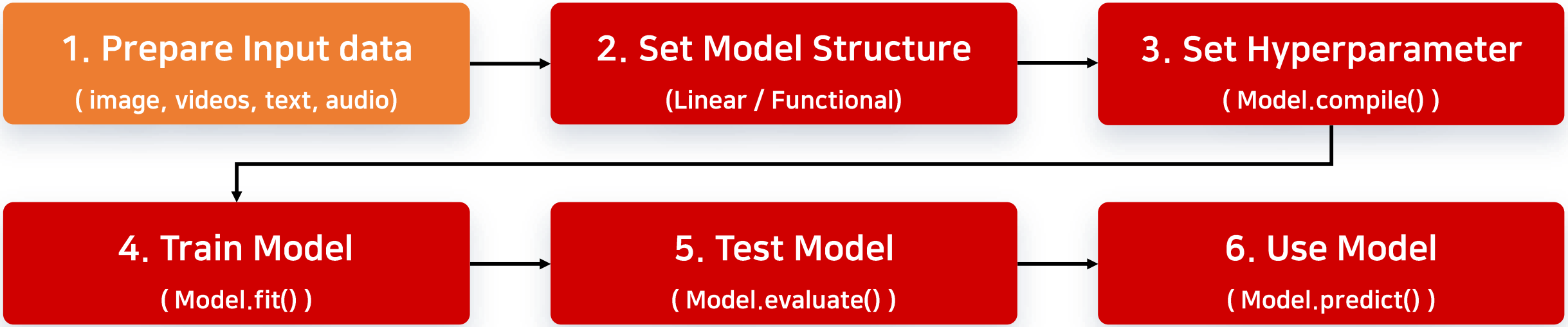
PIAI Research Department

- Keras 주요 학습과정



Keras Framework Flow

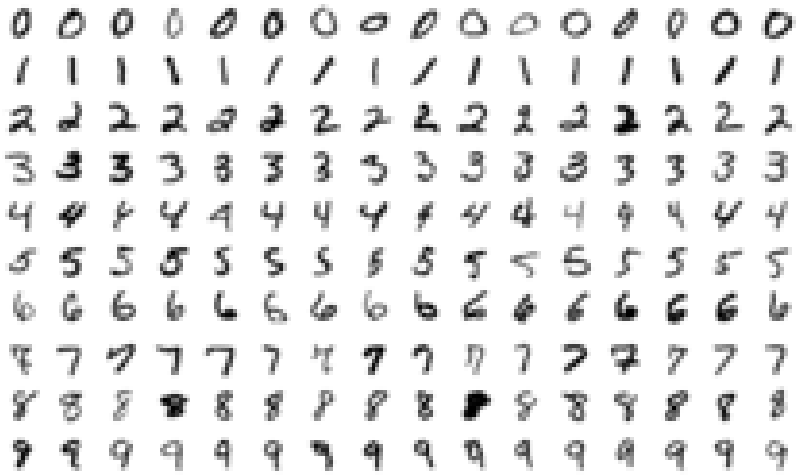
PIAI Research Department



Input Data Preparation

PIAI Research Department

- MNIST Dataset



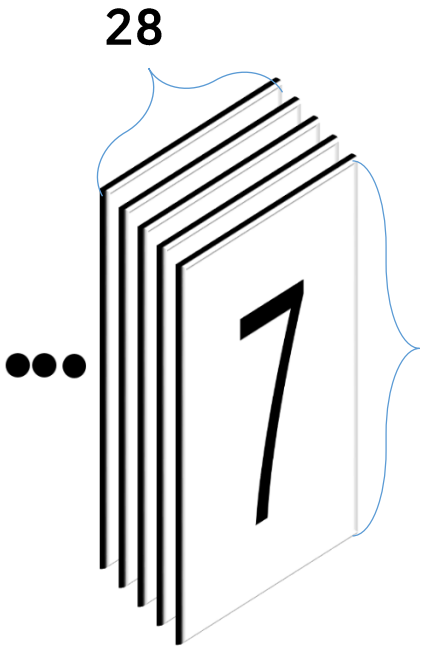
Label	5	0	4	1

- 손으로 쓴 숫자들로 이루어진 데이터셋
- 0~9의 10개 클래스로 구성
- 28X28크기의 grayscale 이미지와 라벨
- Training sample : 60,000개
- Test sample : 10,000개

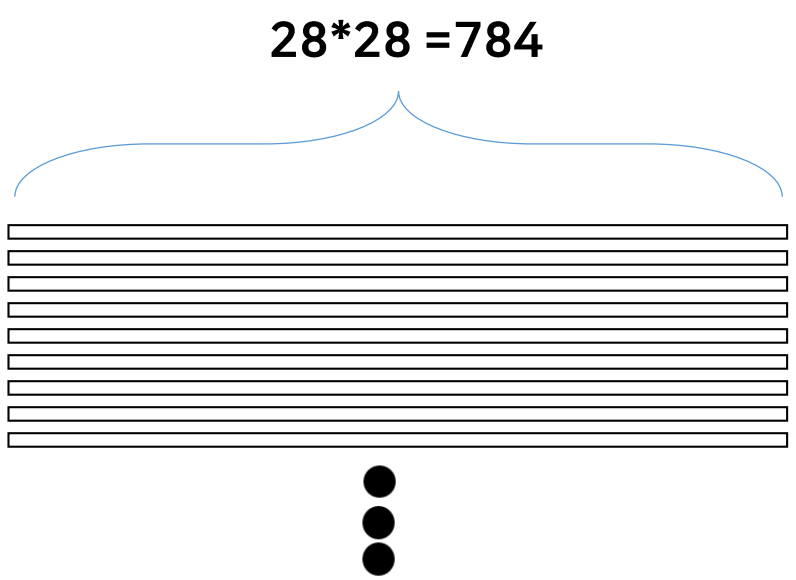
Input Data Preparation

PIAI Research Department

Image



flatten



Label

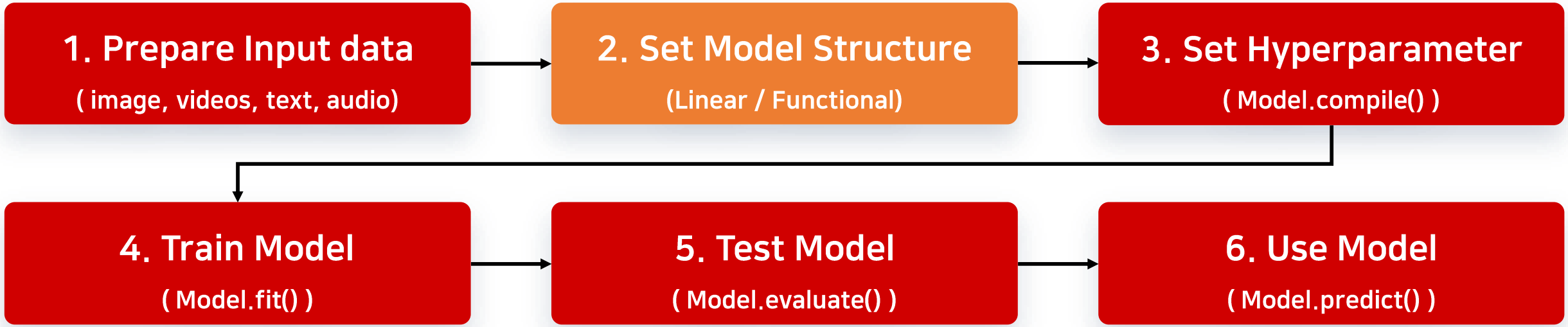
..... 4 5 7

one-hot encoding

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

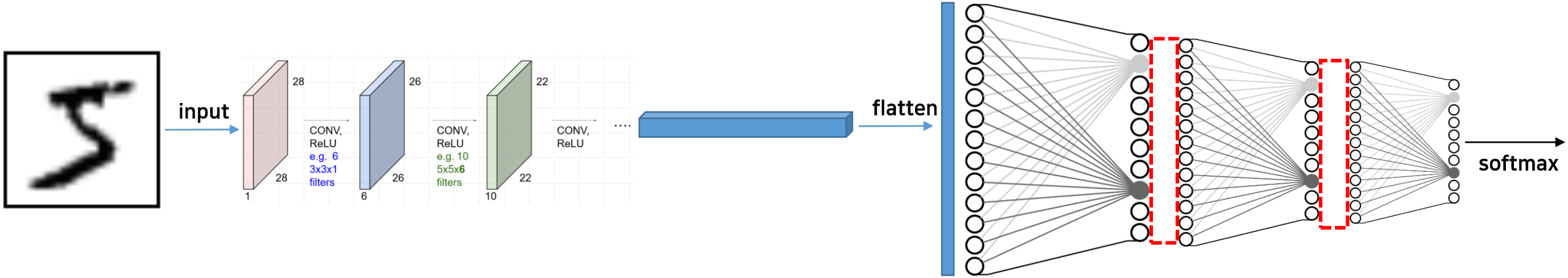
Keras Framework Flow

PIAI Research Department



Overview of Image classification

PIAI Research Department

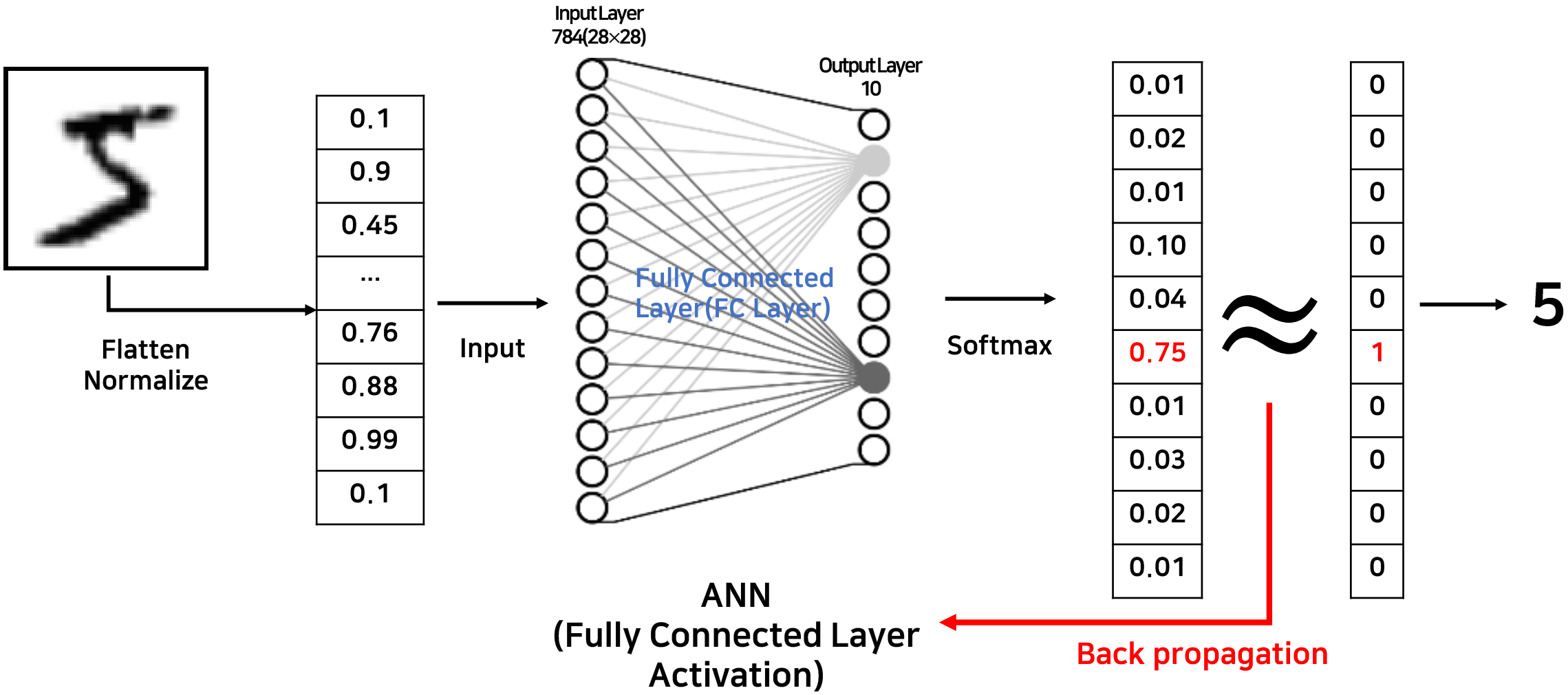


Feature extraction
(Convolution layer
Pooling layer)

Classifier
(Fully Connected layer
Activation layer)

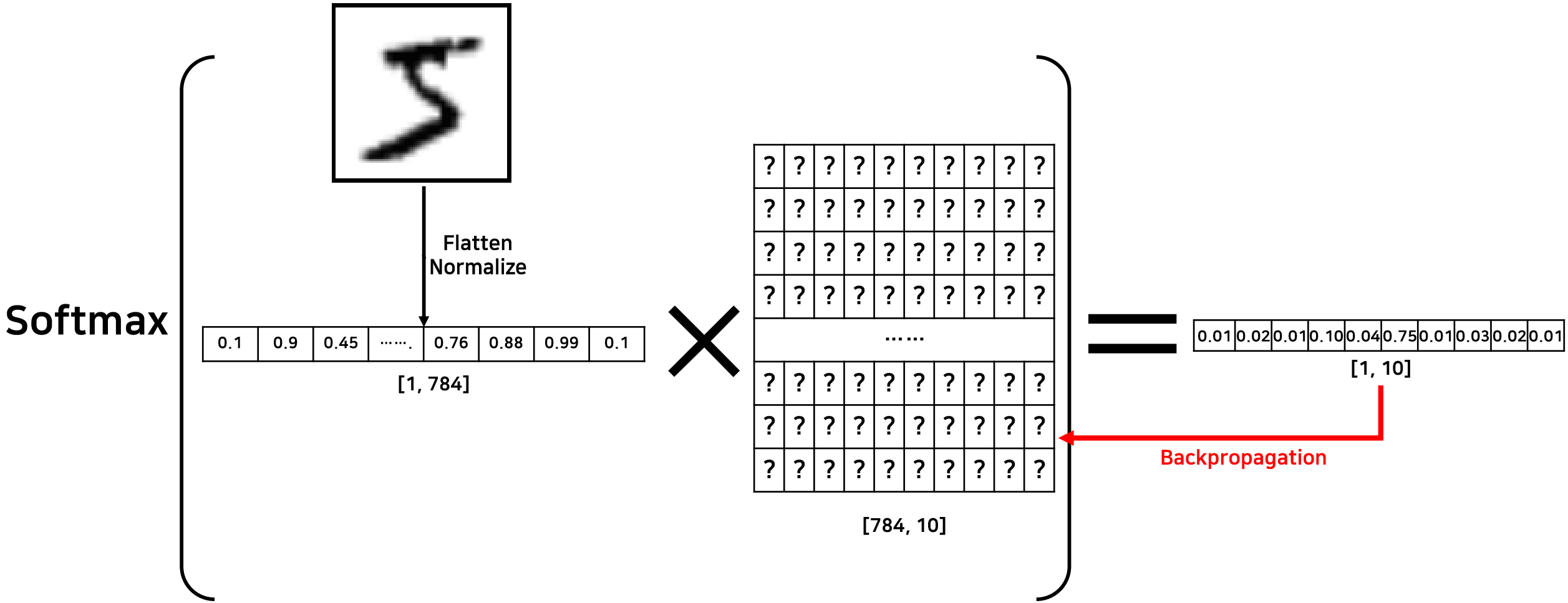
Overview of Classifier

PIAI Research Department



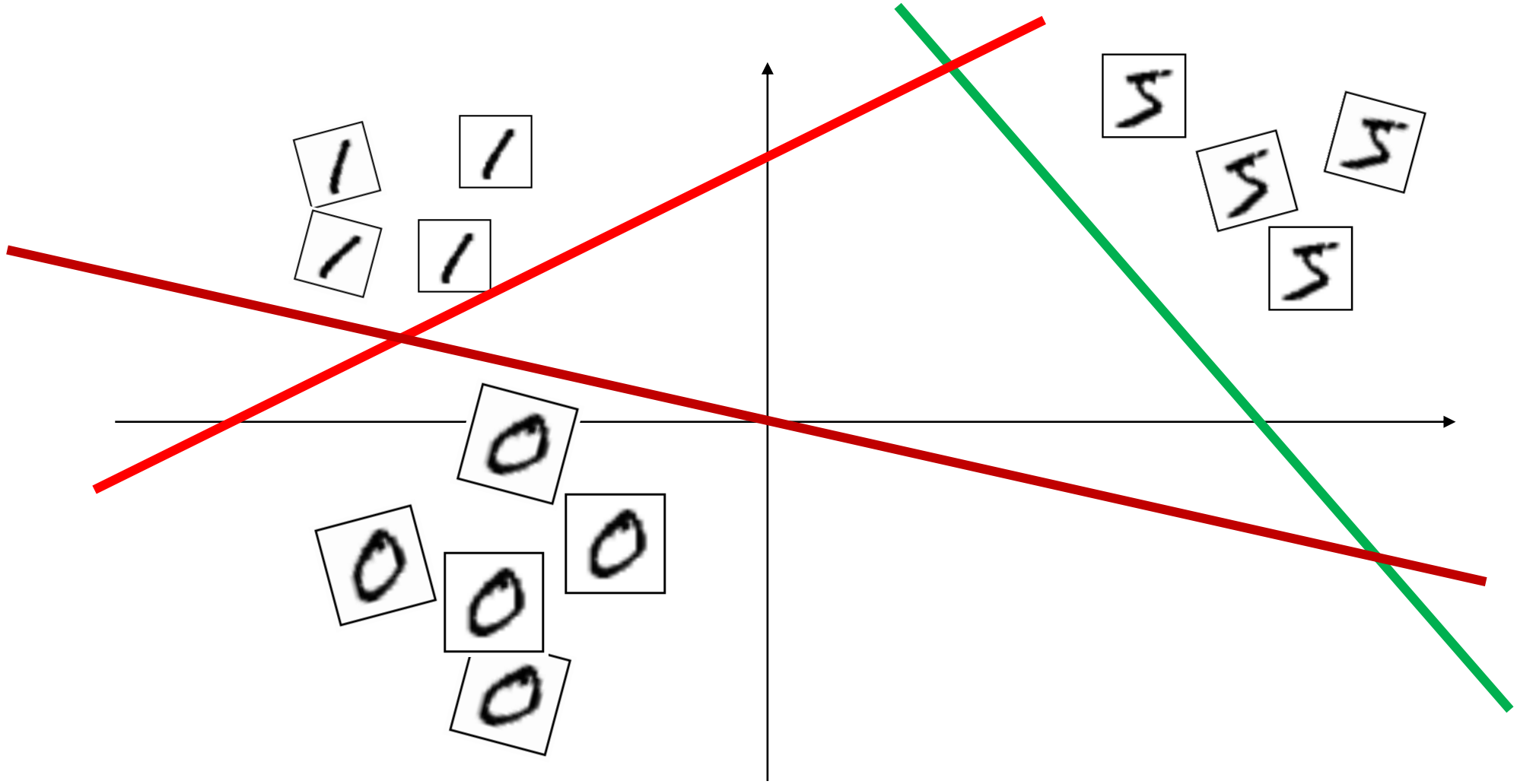
Linear Classifier

PIAI Research Department



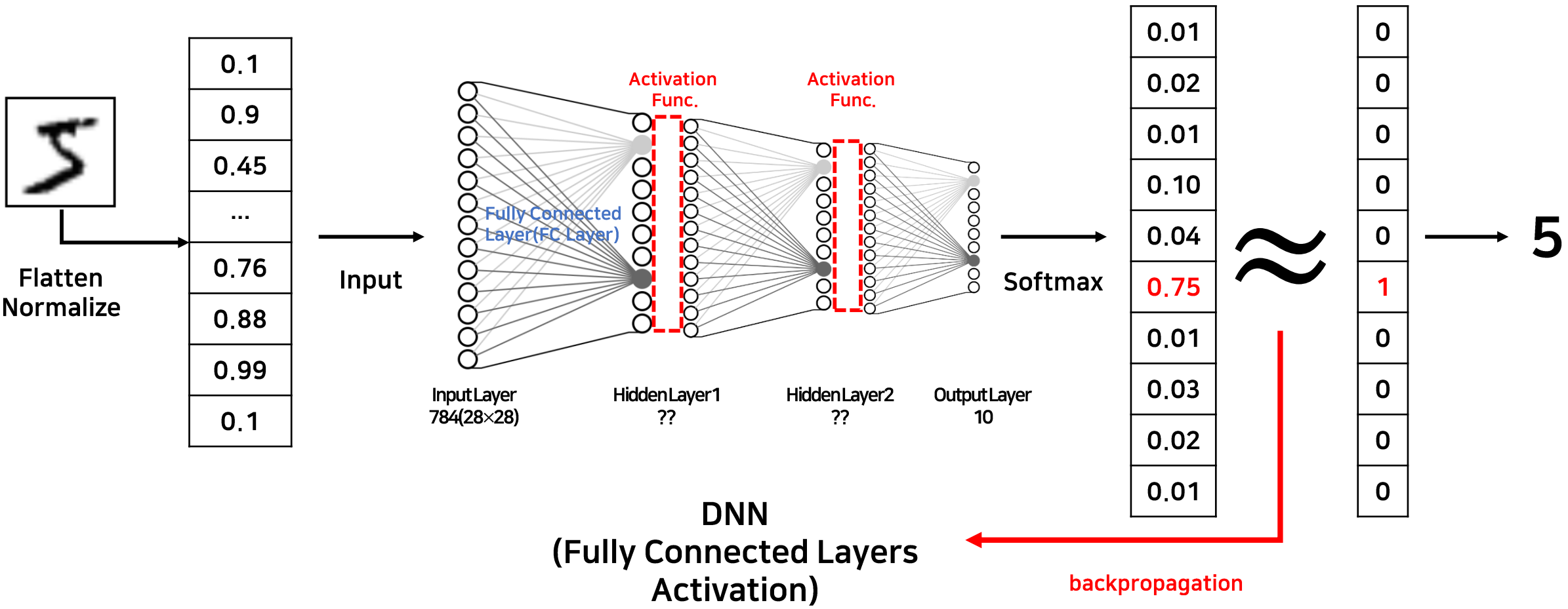
Linear Classifier

PIAI Research Department



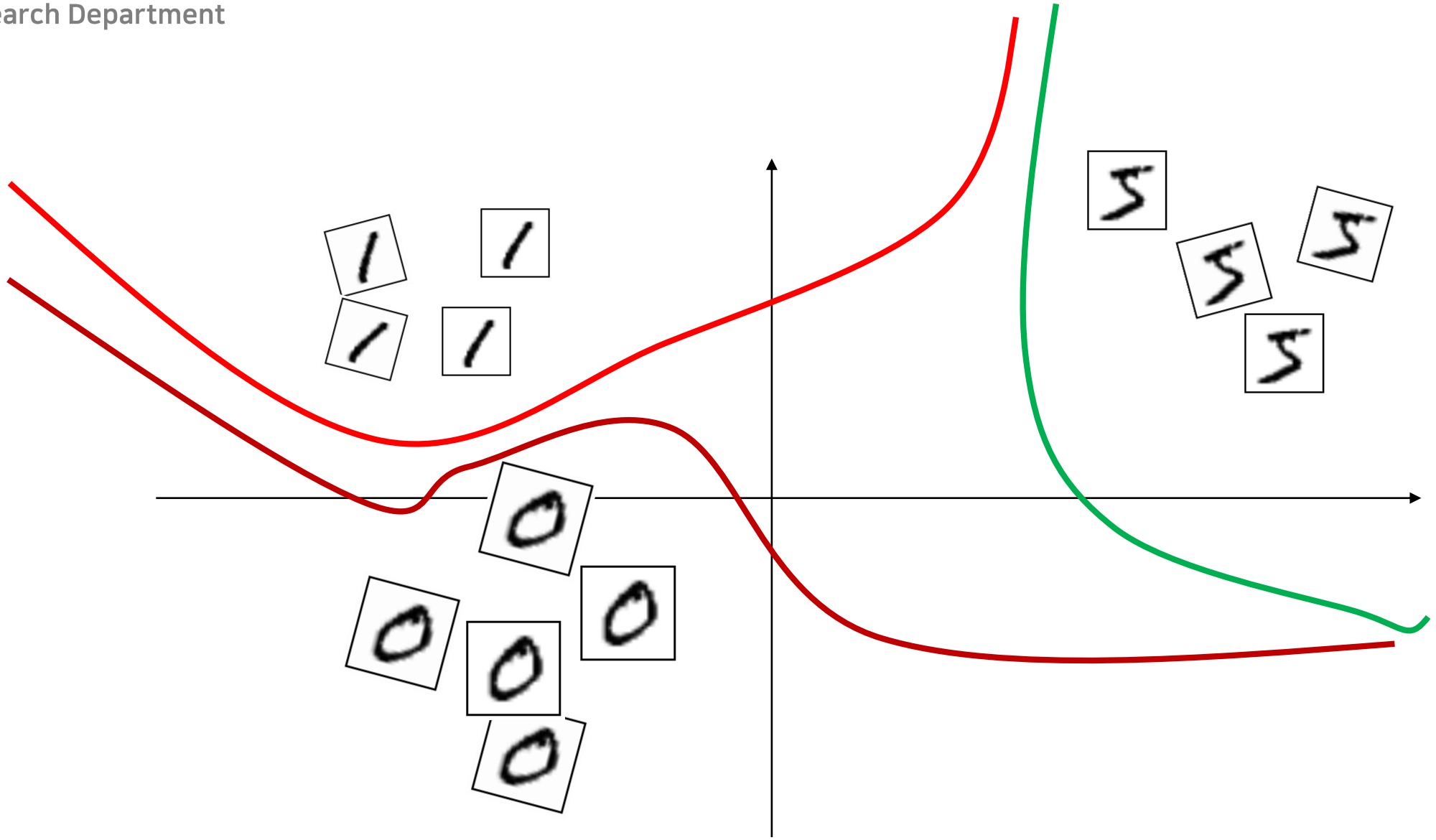
Overview of DNN

PIAI Research Department



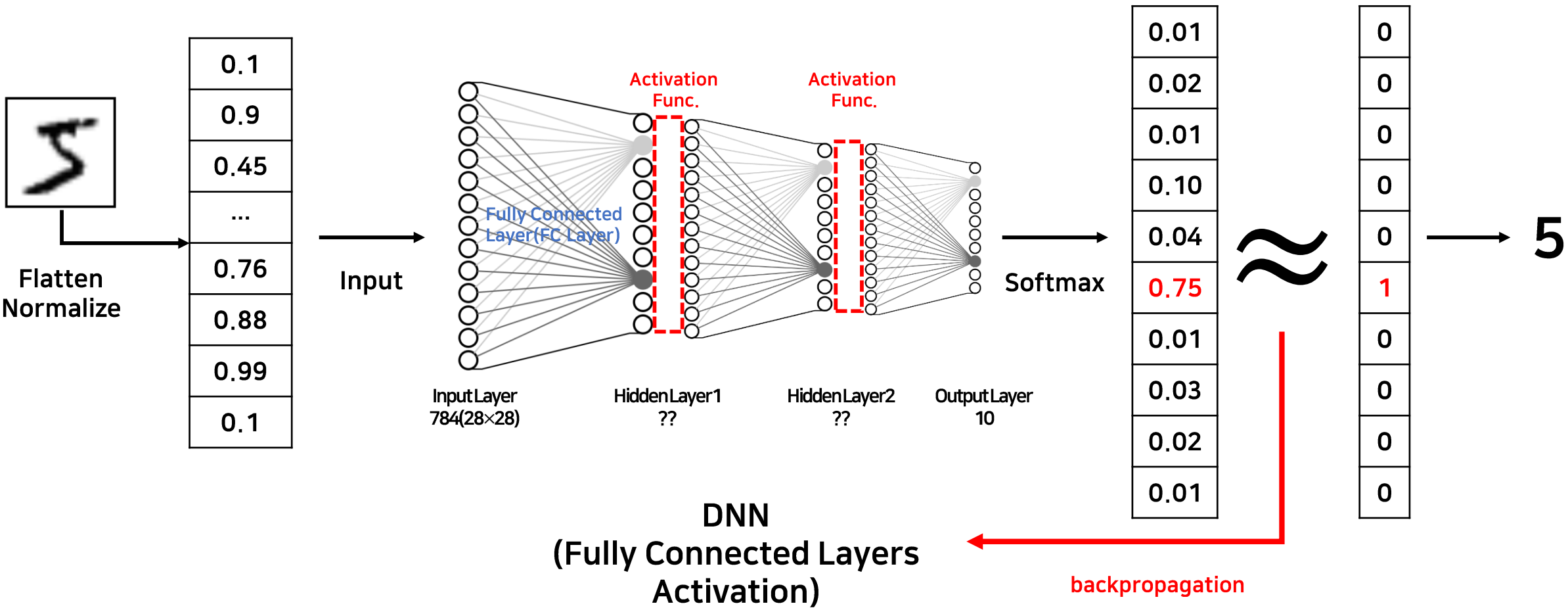
DNN Classifier

PIAI Research Department



Classifier

PIAI Research Department

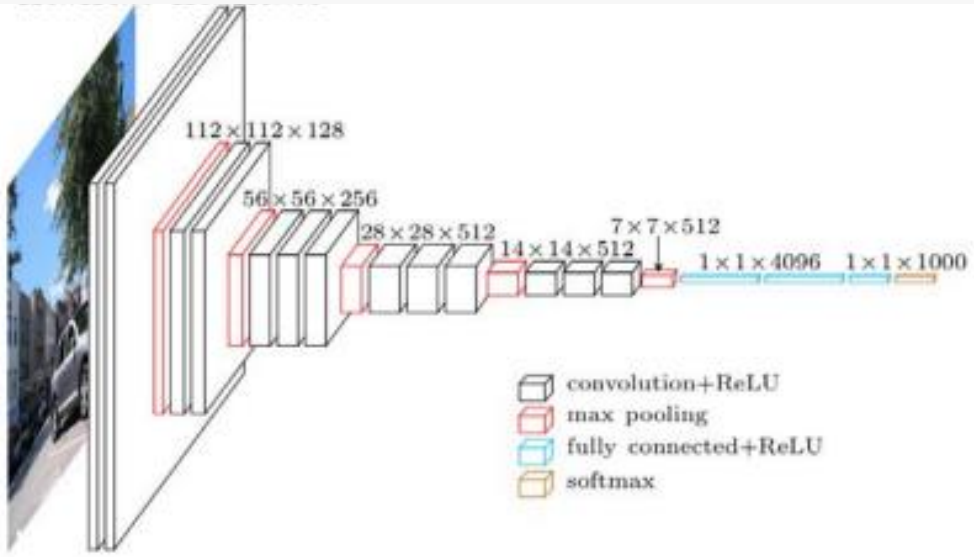


Network Definition

PIAI Research Department

2. Define the model (Sequential style)

```
def DNN_seq(n_in, n_out):
    model = Sequential()
    model.add(Dense(units =256, input_shape=(n_in,), activation='relu'))
    model.add(Dense(units =128, input_shape=(256,), activation='relu'))
    model.add(Dense(units =64, input_shape=(128,), activation='relu'))
    model.add(Dense(units =32, input_shape=(64,), activation='relu'))
    model.add(Dense(units =n_out, input_shape=(32,), activation='softmax'))
    return model
```



- Linear stack of layers
- Useful for building simple models
- Simple classification network
- Encoder – Decoder models

Network Definition

PIAI Research Department

2. Define the model (Sequential style)

```
def DNN_seq(n_in, n_out):
    model = Sequential()
    model.add(Dense(units =256, input_shape=(n_in,), activation='relu'))
    model.add(Dense(units =128, input_shape=(256,), activation='relu'))
    model.add(Dense(units =64, input_shape=(128,), activation='relu'))
    model.add(Dense(units =32, input_shape=(64,), activation='relu'))
    model.add(Dense(units =n_out, input_shape=(32,), activation='softmax'))
    return model
```

- Sequential : Sequential 스타일로 진행하겠다고 설정
- Dense : Fully Connected Layer 종류
- Units : output 원소 개수
- Input_shape : input의 크기
- Activation : activation 함수 설정

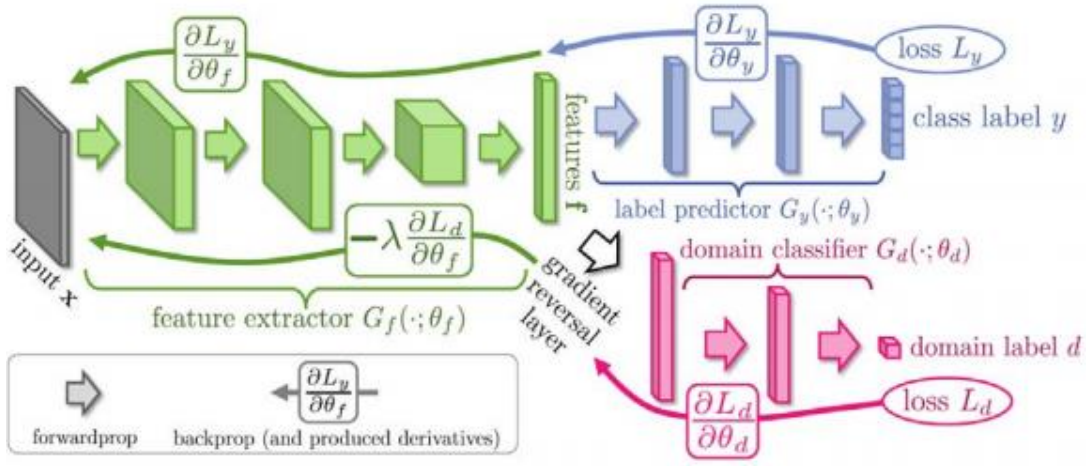
Network Definition

PIAI Research Department

2. Define the model (Functional style)

```
def DNN_func(n_in, n_out):
    x = Input(shape=(n_in,))
    x = Dense(256)(x)
    x = Activation('relu')(x)
    x = Dense(128)(x)
    x = Activation('relu')(x)
    x = Dense(64)(x)
    x = Activation('relu')(x)
    x = Dense(32)(x)
    x = Activation('relu')(x)
    x = Dense(n_out)(x)
    y = Activation('softmax')(x)
    model = Model(inputs = x, outputs = y)
    return model
```

- Multi-input and Multi-output models
- Complex models which forks into 2 or more branches
- Models with shared (Weights) layers



Network Definition

PIAI Research Department

2. Define the model (Functional style)

```
def DNN_func(n_in, n_out):
    x = Input(shape=(n_in,))
    x = Dense(256)(x)
    x = Activation('relu')(x)
    x = Dense(128)(x)
    x = Activation('relu')(x)
    x = Dense(64)(x)
    x = Activation('relu')(x)
    x = Dense(32)(x)
    x = Activation('relu')(x)
    x = Dense(n_out)(x)
    y = Activation('softmax')(x)
    model = Model(inputs = x, outputs = y)
    return model
```

- Input : Input Layer로 정의
- Dense(256)(x) : x layer를 input으로 256크기의 output이 나오도록 Fully Connected layer를 구성
- Activation('relu')(x) : x layer를 입력해 relu 활성화 함수를 적용
- Model(inputs = x, outputs = y) : x를 입력해서 y가 출력될 수 있도록 Model을 정의

Network Definition

PIAI Research Department

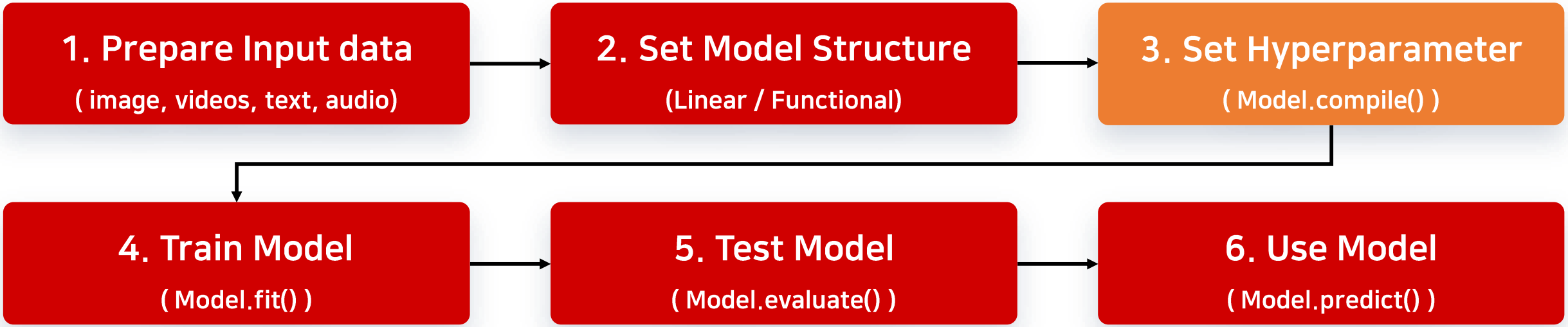
2. Define the model

```
model = DNN_seq(n_in, n_out)
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_17 (Dense)            (None, 256)               200960

 dense_18 (Dense)            (None, 128)               32896

 dense_19 (Dense)            (None, 64)                8256

 dense_20 (Dense)            (None, 32)                2080

 dense_21 (Dense)            (None, 10)                330

=================================================================
Total params: 244,522
Trainable params: 244,522
Non-trainable params: 0
_________________________________________________________________
```

Keras Framework Flow

PIAI Research Department



Model Setting

PIAI Research Department

3. Model.compile() (Set Loss / Optimizer and Metrics)

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

Loss Functions: 알고리즘이 예측한 값과 실제 정답의 차이를 비교하기 위한 함수

종류

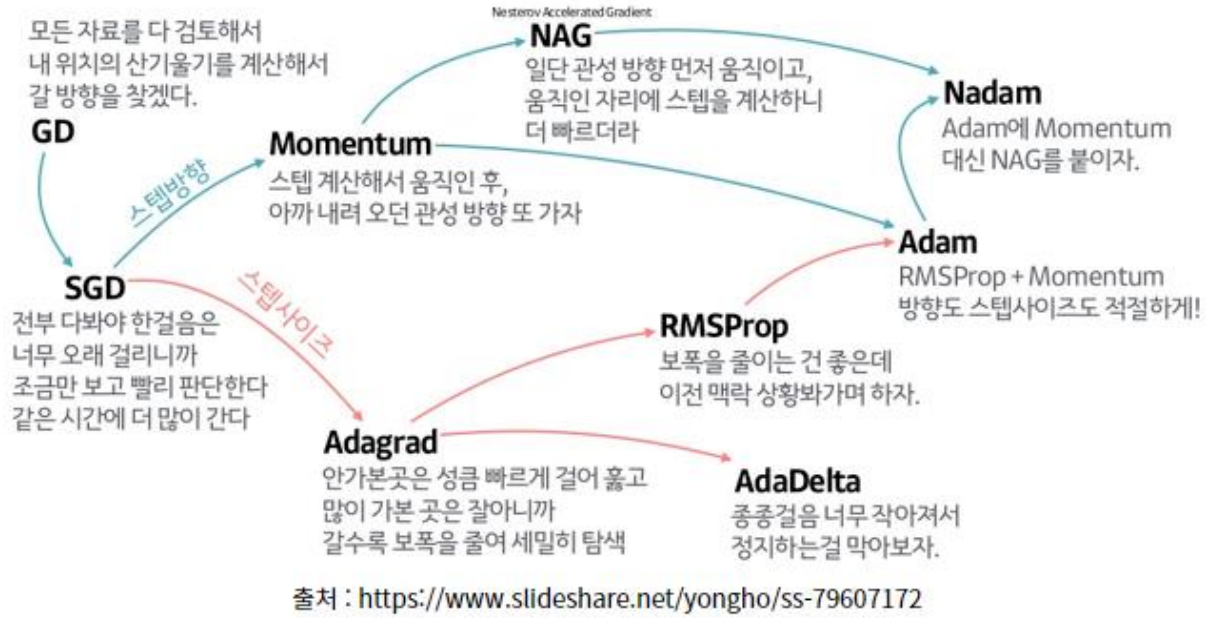
- Probablistic losses: binary_crossentropy, categorical_crossentropy, sparse_categorical_crossentory, poisson, kl_divergence
- Regression loss: rmse, mse, mae, mape, msle, cosine_similarity
- Hinge loss: hinge, squared_hinge, categorical_hinge,

Model Setting

PIAI Research Department

3. Model.compile() (Set optimizer and loss function)

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



출처 : https://www.slideshare.net/yongho/ss-79607172

Optimizer Loss function의 최솟값을 찾아주고자 할 때 수행할 알고리즘

종류 : sgd, RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam, Ftrl

Model Setting

PIAI Research Department

3. Model.compile() (Set optimizer and loss function)

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

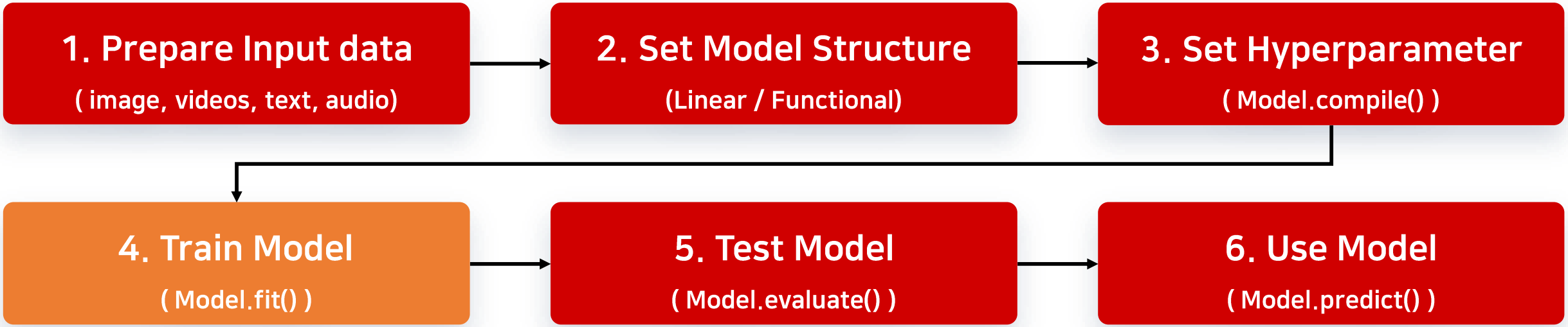
Metrics : 검증 데이터에서의 학습 모델의 성능에 대한 평가 지표 설정

종류:

- Loss
- accuracy, top_k_categorical_accuracy
- Recall, Precision, AUC
- MeanIoU

Keras Framework Flow

PIAI Research Department



Model Training

PIAI Research Department

4. Model.fit() (Train model)

```
model.fit(train_X, train_Y, epochs=epochs, batch_size=batch_size, validation_data=(vali_X, vali_Y), shuffle=False, callbacks=[checkpointer])

checkpointer=ModelCheckpoint(filepath=modelpath, monitor='val_acc', verbose=0, save_best_only=True)
```

- Train_X: Training data
- Train_Y: Target label
- validation_data: Tuple (vali_x, vali_y)
- validation_split:Fraction of the training data to be used as validation data (0~1 사이값)
- shuffle: whether to shuffle the training data before each epoch (Boolean)
- verbose: Verbosity mode(0, 1, or 2)

Model Training

PIAI Research Department

4. Model.fit() (Train model)

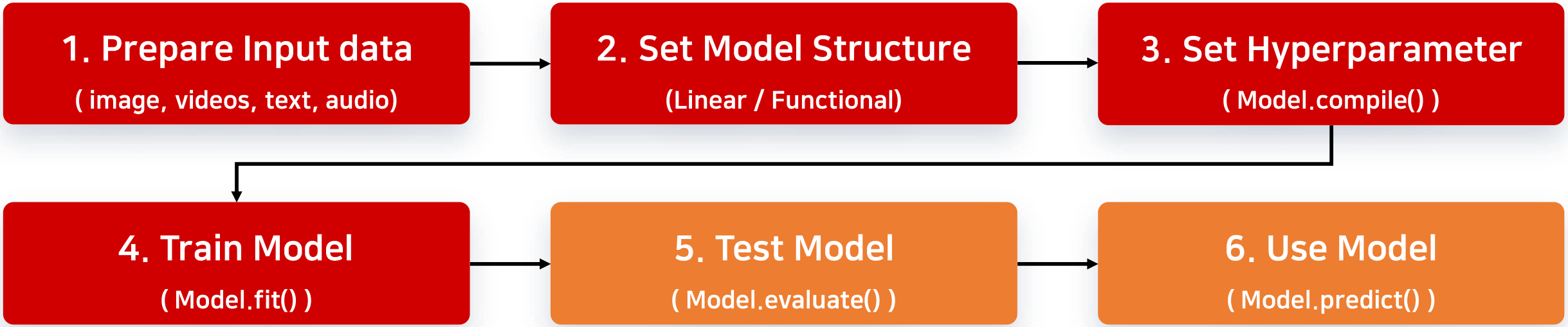
```
model.fit(train_X, train_Y, epochs=epochs, batch_size=batch_size, validation_data=(vali_X, vali_Y), shuffle=False, callbacks=[checkpointer])

checkpointer=ModelCheckpoint(filepath=modelpath, monitor='val_acc', verbose=0, save_best_only=True)
```

- epochs: Number of epochs to train the model
- batch_size: Number of samples per gradient update

- 1 Epoch : 전체 데이터 셋에 대해 한번 학습을 완료한 상태를 의미(보통 1 Epoch 후 validation 진행)
- Iteration : 1 Epoch을 완료하기 위해 필요한 Pass 회수(batch_size × Iteration ≈ Dataset Size)

Keras Framework Flow

PIAI Research Department



Model Application

PIAI Research Department

5. Model.evaluate() (Test model)

```
loss_and_accuracy = model.evaluate(X_test, Y_test, batch_size=128)
print('loss : %.4f, accruracy : %.4f'%(loss_and_accuracy[0],loss_and_accuracy[1]))
```

- x: test data
- y: target label
- batch_size: Number of samples per evaluation step
- verbose: Verbosity mode

6. Model.predict() (Use model)

```
model.predict(x, batch_size=128)
```

- x: input data
- batch_size: Number of samples per predict
- verbose: Verbosity mode

Code Running

(student) 1_Keras Basic & Classifier.ipynb

PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

PyTorch Basic

I-Eon Na

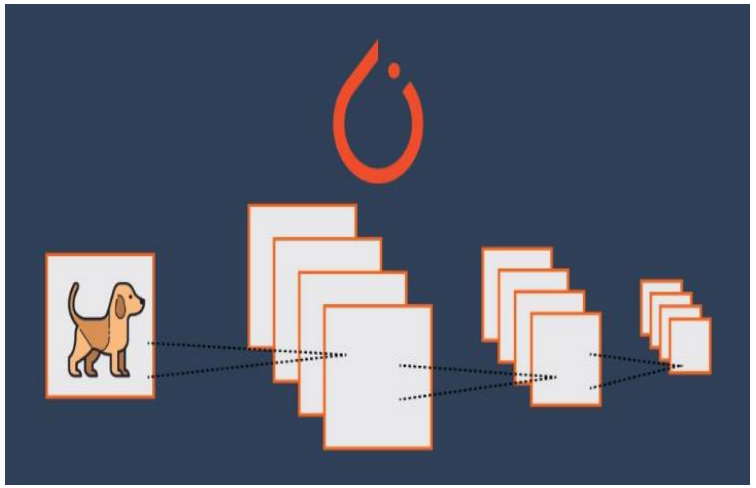
PyTorch Tutorial

PIAI Research Department

PYTORCH

- PyTorch (torch.nn) module
- Base on Torch and Caffe2
- similar Deep Learning library to TensorFlow

- Made by Facebook AI Research Team
- Simple to make and edit network and parameters on GPU



Keras VS PyTorch

Keras

간단..?

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='
model.add(MaxPool2D())
model.add(Conv2D(16, (3, 3), activation='
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(10, activation='softmax')
```

동일 로직, 다른 코드

- 데이터 전처리 과정 간편
- 단일 코드로 간단한 모델 제작 가능
- Verbose 기능 제공(학습 추이 출력)

PyTorch

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(3, 32
        self.conv2 = nn.Conv2d(32, 1
        self.fc1 = nn.Linear(16 * 6
        self.pool = nn.MaxPool2d(2,

    def forward(self, x):
        x = self.pool(F.relu(self.co
        x = self.pool(F.relu(self.co
        x = x.view(-1, 16 * 6 * 6)
        x = F.log_softmax(self.fc1(x

        return x
```

복잡..?

```
model = Net()
```

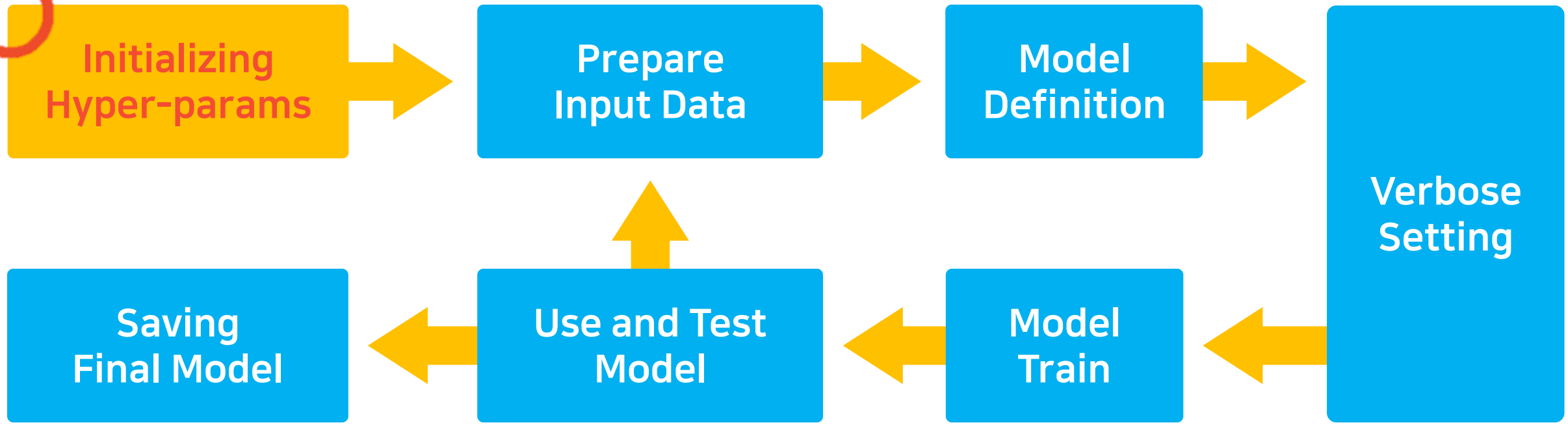
- 데이터 전처리 시 '데이터 로더' 개념 숙지 필요
- 코드가 비교적 복잡하지만 세부 파라미터 수정에 대한 자유도가 높음
- Verbose 기능 직접 코딩

출처: https://github.com/KerasKorea/KEKOxTutorial/blob/master/42_keras_or_pytorch_as_your_first_deep_learning_framework.md

PyTorch Framework Flow

PIAI Research Department

- Deep Learning 모델 구축 과정



HyperParameter Setting

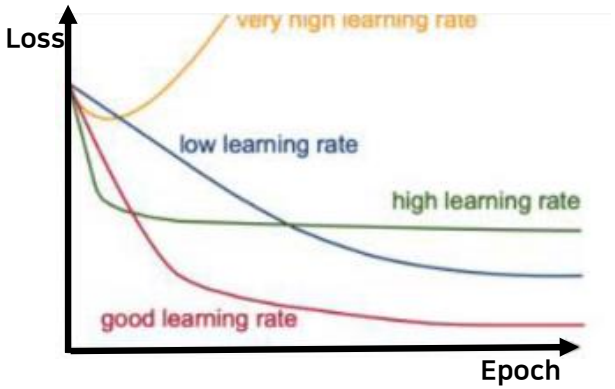
PIAI Research Department

- **하이퍼 파라미터 설정 - 1**

```
# 데이터 로더
batch_size = 32
```

```
# 하이퍼 파라미터
maximum_epoch = 10 # 최대 epoch
learning_rate = 0.01 # Learning Rate
```

- 데이터를 한번에 몇 개를 넣을지
- 전체 데이터셋을 몇 번 학습할지
- 학습하는 속도 조절



출처: https://bdtechtalks.com/2020/04/27/deep-learning-mode-connectivity-adversarial-attacks/gradient-descent-local-minima/
출처: https://m.blog.naver.com/qbxlvnf11/221449297033

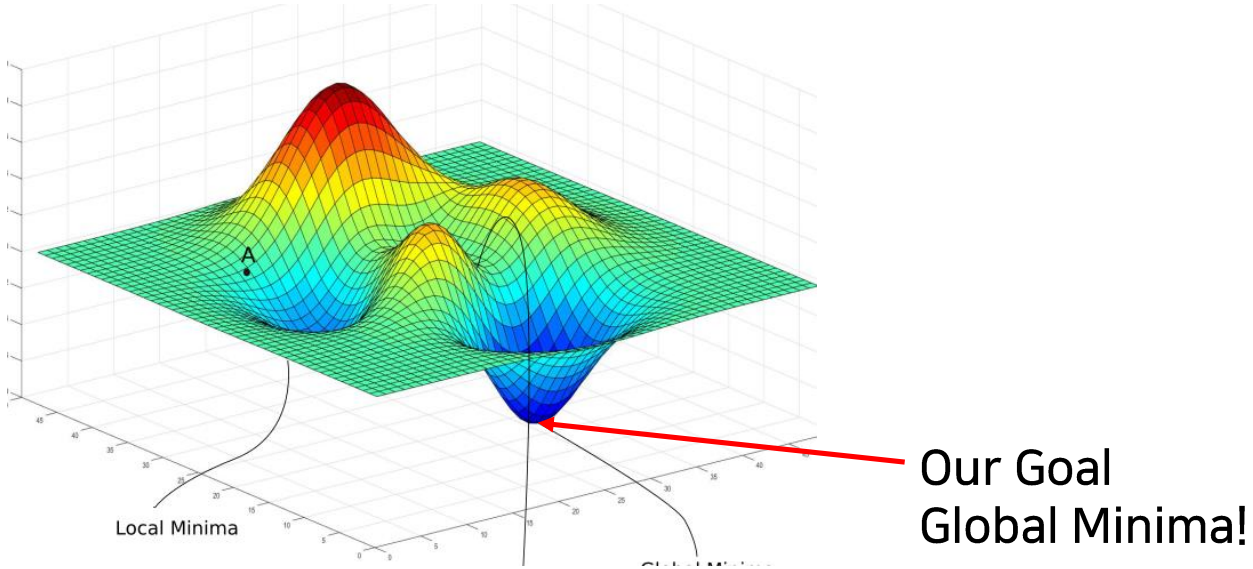
HyperParameter Setting

PIAI Research Department

- **하이퍼 파라미터 설정 - 2**

```
1  # Model
2  def init_model(_net):
3      global net, loss_fn, optim
4      net = _net.to(device)
5      loss_fn = nn.CrossEntropyLoss()
6  #     optim = Adam(net.parameters(), lr=learning_rate)
7      optim = SGD(net.parameters(), lr=learning_rate)
```

- 정답지와 예측값을 비교하여 '모델이 틀린 정도 ' 에 대해서 알아내는 Loss Function
- Gradient Descent를 어떻게 할지에 대해 Optimizer를 지정 (Stochastic Gradient Descent)



Our Goal
Global Minima!

출처: https://www.researchgate.net/figure/a-Graphical-depiction-of-training-loss-with-varying-learning-rates-The-optimal_fig7_346206645

HyperParameter Setting

PIAI Research Department

- **PyTorch에서 사용 가능한 Optimizer 종류**

`Adadelta`	Implements Adadelta algorithm.
`Adagrad`	Implements Adagrad algorithm.
`Adam`	Implements Adam algorithm.
`AdamW`	Implements AdamW algorithm.
`SparseAda`	Implements lazy version of Adam algorithm suitable for sparse tensors.
`Adamax`	Implements Adamax algorithm (a variant of Adam based on infinity norm).
`ASGD`	Implements Averaged Stochastic Gradient Descent.
`LBFGS`	Implements L-BFGS algorithm, heavily inspired by minFunc.
`NAdam`	Implements NAdam algorithm.
`RAdam`	Implements RAdam algorithm.
`RMSprop`	Implements RMSprop algorithm.
`Rprop`	Implements the resilient backpropagation algorithm.
`SGD`	Implements stochastic gradient descent (optionally with momentum).

출처: https://pytorch.org/docs/stable/optim.html

HyperParameter Setting

- PyTorch에서 사용 가능한 Loss Function 종류

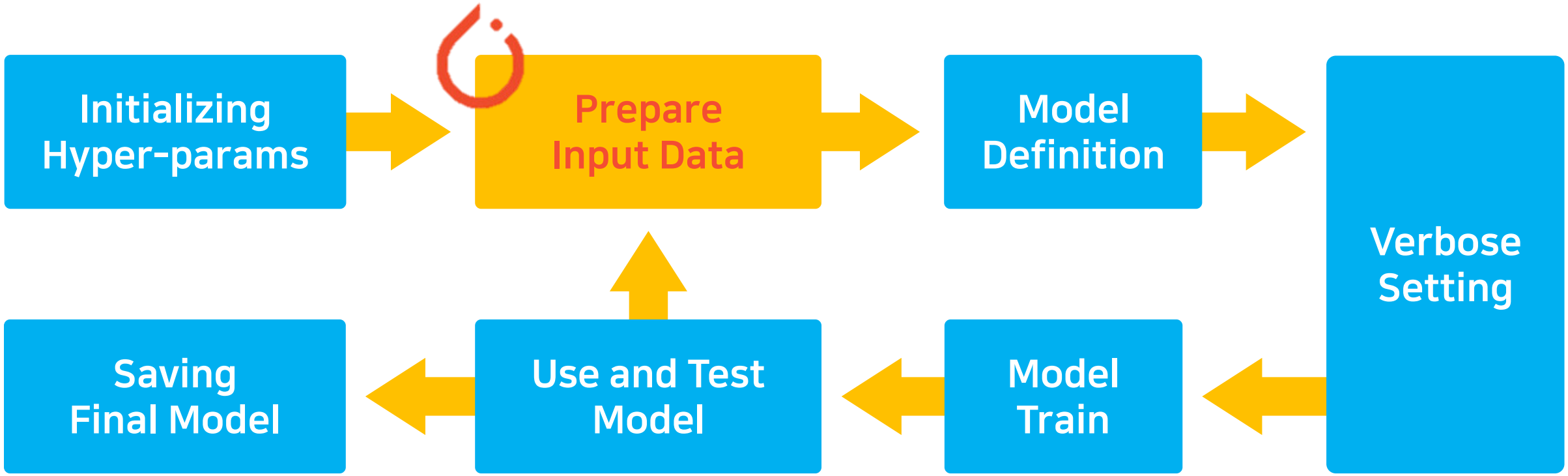
nn.L1Loss
nn.MSELoss
nn.CrossEntropyLoss
nn.CTCLoss
nn.NLLLoss
nn.PoissonNLLLoss
nn.GaussianNLLLoss
nn.KLDivLoss
nn.BCELoss
nn.BCEWithLogitsLoss
nn.MarginRankingLoss
nn.HingeEmbeddingLoss
nn.MultiLabelMarginLoss
nn.HuberLoss
nn.SmoothL1Loss
nn.SoftMarginLoss
nn.MultiLabelSoftMarginLoss
nn.CosineEmbeddingLoss
nn.MultiMarginLoss
nn.TripletMarginLoss
nn.TripletMarginWithDistanceLoss

출처: https://pytorch.org/docs/stable/nn.html#loss-functions

PyTorch Framework Flow

PIAI Research Department

- Deep Learning 모델 구축 과정



Dataset Preparation

- **데이터셋 Tensor 변환 & 전처리**

```
1 mnist_transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize(mean=(0.5,), std=(1.0,))])
2
```

텐서(Tensor)라고 하면 텐서플로우를 떠올리기 쉽지만, 사실 딥러닝에서는 데이터를 처리하기 위한 데이터의 형태라고 이해하면 된다.

```
In [355]:
1 a = [1, 2, 3], [4, 5, 6]

In [356]:
1 print(a)

([1, 2, 3], [4, 5, 6])

In [357]:
1 a = torch.tensor(a)
2 print(a)

tensor([[1, 2, 3],
        [4, 5, 6]])
```

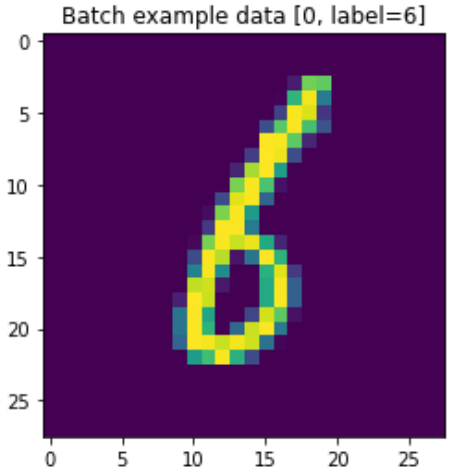
- Transform 옵션에서 텐서화, 정규화, 데이터 어그멘테이션 등 데이터 전처리에 대한 여러가지 옵션 제공
- 딥러닝 학습을 위해서는, Tensor 형식으로 변환해주어야 PyTorch에서 처리 가능

Dataset Preparation

- **전처리된 데이터셋을 DataLoader에 업로드**

```
# Train / Validation / Test 셋에 대한 데이터 로더 만들기
train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True, pin_memory=True, drop_last=True)
valid_loader = DataLoader(valid_data, batch_size=len(valid_data), pin_memory=True)
test_loader = DataLoader(test_data, batch_size=len(test_data), pin_memory=True)
```

```
Idx: 0 / X.shape = torch.Size([32, 1, 28, 28]) / Y.shape = torch.Size([32])

Y[0:32] = tensor([6, 0, 8, 5, 5, 2, 6, 3, 3, 7, 4, 6, 2, 4, 8, 1, 8, 6, 7, 8, 2, 6, 8, 9,
        9, 1, 8, 3, 7, 9, 5, 1])
```



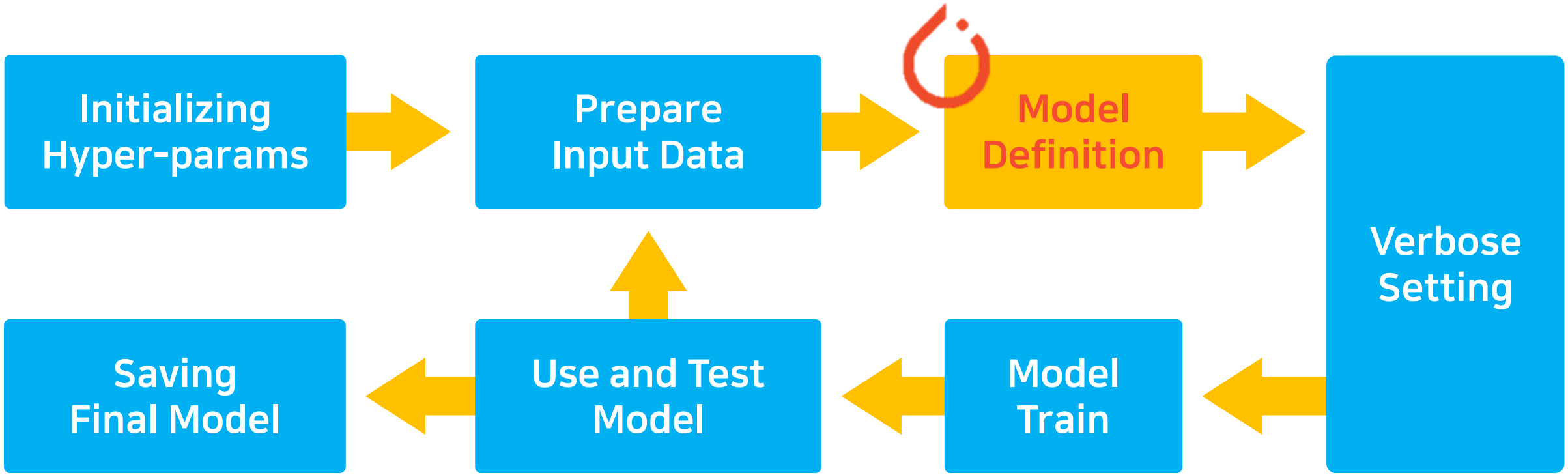

```
# 데이터 로더
batch_size = 32
```

- 전처리된 데이터를 DataLoader에 추가
- DataLoader 출력시, Batchsize에 맞게 구성되어 있음을 확인

PyTorch Framework Flow

PIAI Research Department

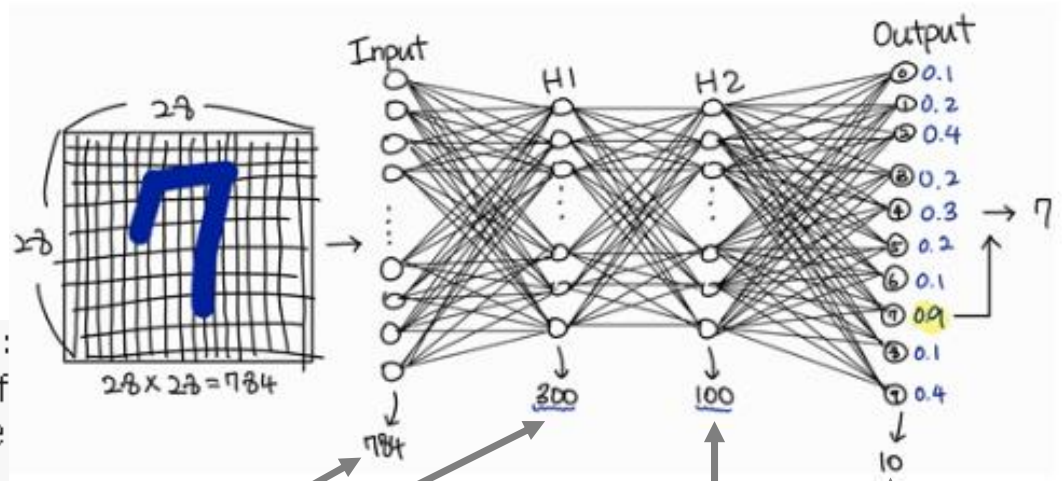
- Deep Learning 모델 구축 과정



Network Architecture Design

PIAI Research Department

- **딥러닝 모델 구조 설계**




```
class MLP(nn.Module):
    def __init__(self
        super(MLP, se

        # 784 -> 300 -> 100 -> 10
        self.fc1 = nn.Linear(784, 300)
        self.act1 = nn.ReLU() # 활성함수 Relu 사용
        self.fc2 = nn.Linear(300, 100)
        self.act2 = nn.ReLU()
        self.fc3 = nn.Linear(100, 10)

    def forward(self, x):
        x = self.act1(self.fc1(x))
        x = self.act2(self.fc2(x))
        x = self.fc3(x)
        return x
```

- MLP(Multi Layer Perceptron) 모델 구조 설계

  1) 28 * 28의 pixel로 구성된 사진을 784개의 뉴런으로 입력

  2) 입력단과 출력단 사이에 HL 추가 (정확도 ↑ but 연산량 ↑)

  3) 히든 레이어에는 Activation Func.으로 ReLu 사용

  4) 입력을 0 ~ 9 中 하나로 분류하기 위하여 10개의 class로 결과 표출

Training Algorithm Setting

PIAI Research Department

```
def epoch(data_loader):
    global epoch_cnt
    iter_loss, iter_acc = [], []
    last_grad_performed = False

    # Mini-batch iterations
    for _data, _label in data_loader:
        data, label = _data.view([len(_data), -1]).tc


        # 1. Feed-forward
        result = net(data)
        _, out = torch.max(result, 1)

        # 2. Calculate loss
        loss = loss_fn(result, label)
        iter_loss.append(loss.item())

        # 3. Backward propagation(train mode 에서만)
        if result.requires_grad:
            optim.zero_grad()
            loss.backward()
            optim.step()
            last_grad_performed = True

        # 4. Calculate accuracy
        acc_partial = (out == label).float().sum()
        acc_partial = acc_partial / len(label)
        iter_acc.append(acc_partial.item())
```

Validation/Test 경우 torch.no_grad() 를 사용하여 if문을 만족 -> 학습 안함

```
with torch.no_grad():
    vloss, vacc = epoch(valid_loader)
```

- 모델이 Loss를 줄이는 방향으로 학습하도록 알고리즘 구축

1) 모델에 데이터 투입

2) 정답지와 결과를 비교하여 Loss 산정(Loss Function 사용)

3) Loss를 줄이기 위하여 Back Pro -pagation시행(Train 에서만)

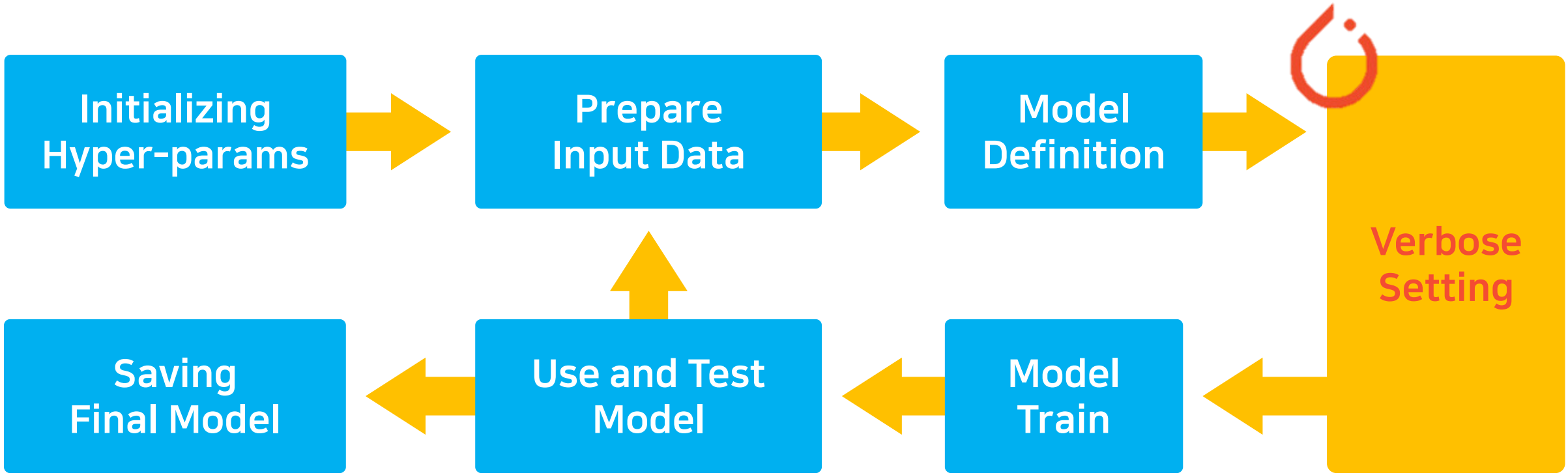
4) 1 Step 만큼 Gradient Decent 진행(Learning Rate만큼 Activ -ation Function으로 하강)

5) 1 ~ 4번을 Epochs 만큼 반복

PyTorch Framework Flow

PIAI Research Department

- Deep Learning 모델 구축 과정



Verbose Setting

- **Verbose(학습 Log 출력 여부) 설정**

```
def record_train_log(_tloss, _tacc, _time):
    # Train Log를 추가해준다
    time_log.append(_time)
    tloss_log.append(_tloss)
    tacc_log.append(_tacc)
    iter_log.append(epoch_cnt)

def record_valid_log(_vloss, _vacc):
    # Validation Log를 추가해준다
    vloss_log.append(_vloss)
    vacc_log.append(_vacc)
```

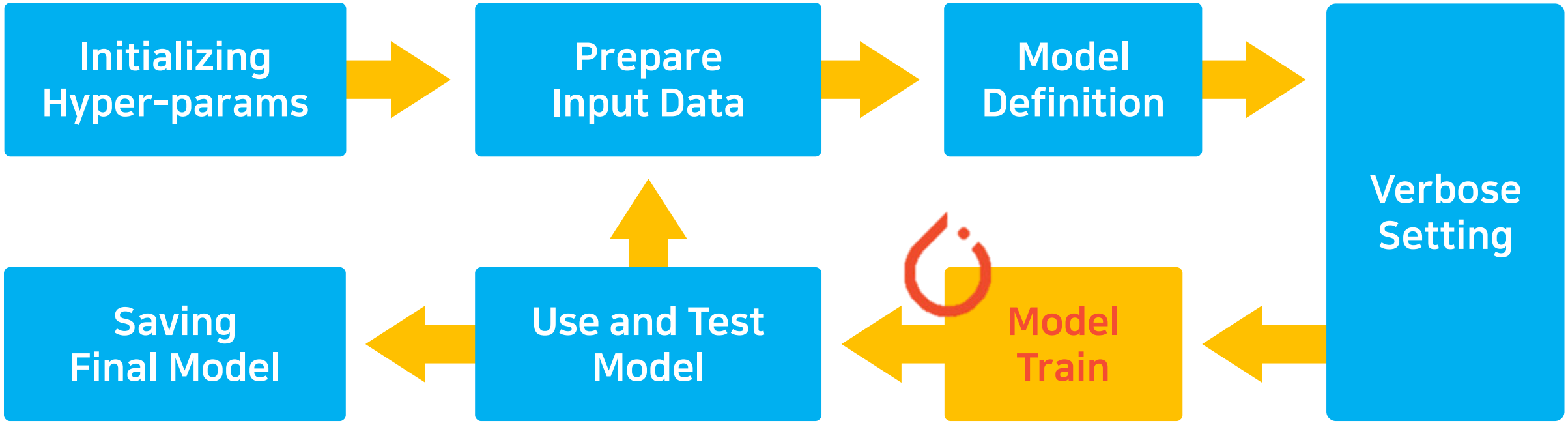
```
def print_log():
    # 로그 프린트

    # 소숫점 3자리 수까지 조절
    train_loss = round(last(tloss_log), 3)
    train_acc = round(last(tacc_log), 3)
    val_loss = round(last(vloss_log), 3)
    val_acc = round(last(vacc_log), 3)
    time_spent = round(last(time_log), 3)

    log_str = 'Epoch: {:3} | T_Loss {:5} | T_acc {:5} | V_Loss {:5} | V_acc. {:5} | W
    {:5}'.format(last(iter_log), train_loss, train_acc, val_loss, val_acc, time_spent)

    log_stack.append(log_str) # 프린트할 리스트에 추가
    clear_output(wait=True) # 출력물 모두 초기화

    for log_print in log_stack:
        print(log_print) # 매 에폭마다 추이를 출력
```

- 모델이 학습되는 동안의 실시간 추이를 볼 수 있도록 알고리즘 구축
- Train과 Validation의 Loss와 Acc.를 기록하는 기능
- 매 Epoch 마다 추이를 출력

PyTorch Framework Flow

PIAI Research Department

- Deep Learning 모델 구축 과정



Model Training

PIAI Research Department

- 딥러닝 모델 학습

```
1  # Training Initialization
2  init_model(ANN(784, 10))
3  init_epoch()
4  init_log()
5  losses_train, losses_val = [], []
6
7  # Training Iteration
8  while epoch_not_finished():
9      start_time = time.time()
10
11     tloss, tacc = epoch(train_loader)
12
13     losses_train.append(tloss)
14     end_time = time.time()
15     time_taken = end_time - start_time
16     record_train_log(tloss, tacc, time_taken)
17
18     with torch.no_grad():
19         vloss, vacc = epoch(valid_loader)
20
21     losses_val.append(vloss)
22     record_valid_log(vloss, vacc)
23     print_log()
24
25 print('\n Training completed!')
```

Verbose 함수

- 설계한 모델과 학습 알고리즘을 불러와 학습 진행
- 모델, Epoch 카운터, 로깅 관련 변수들 초기화
- 설정한 Epoch 수만큼 while(for)문을 돌면서 train 반복
- Train 파트
- Validation 파트
  torch.no_grad() = 역전파 학습을 하지 않는다
- Verbose(학습 추이) 출력

Model Training

PIAI Research Department

- **딥러닝 모델의 학습 추이 출력**

```
    print_log() # 로그를 프린트
print('₩n Training completed!')
```

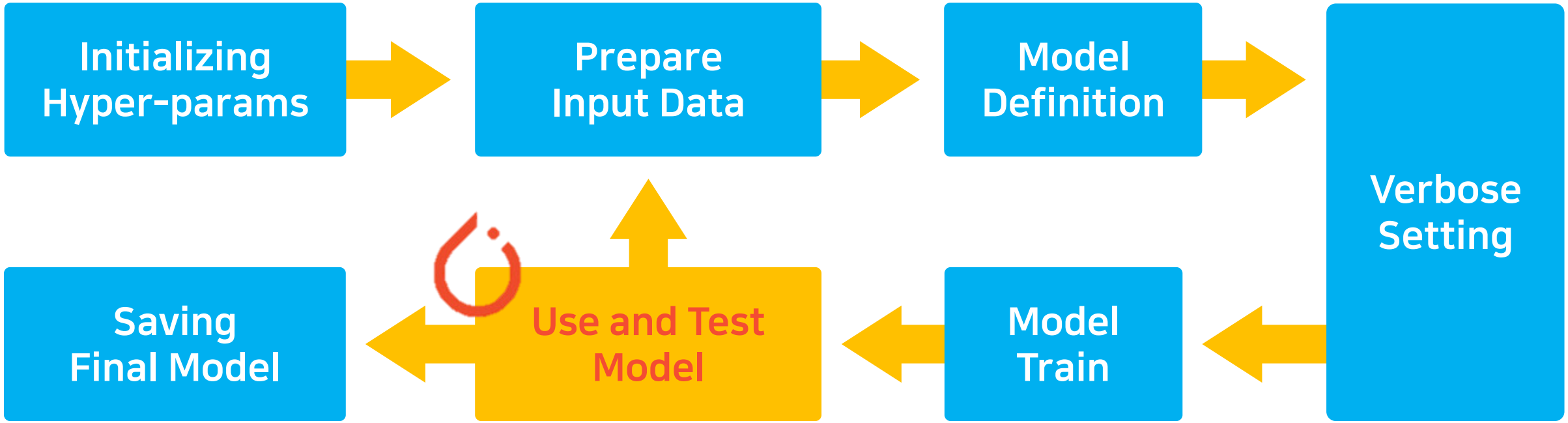
```
Epoch:   1 | T_Loss 1.159 | T_acc 0.704 | V_Loss  0.47 | V_acc. 0.869 | 14.727
Epoch:   2 | T_Loss 0.393 | T_acc 0.887 | V_Loss 0.354 | V_acc. 0.897 | 15.019
Epoch:   3 | T_Loss 0.325 | T_acc 0.905 | V_Loss 0.313 | V_acc.  0.91 | 13.68
Epoch:   4 | T_Loss 0.289 | T_acc 0.915 | V_Loss 0.283 | V_acc. 0.915 | 14.157
Epoch:   5 | T_Loss  0.26 | T_acc 0.924 | V_Loss 0.272 | V_acc. 0.919 | 14.035
Epoch:   6 | T_Loss 0.236 | T_acc 0.931 | V_Loss 0.241 | V_acc.  0.93 | 14.326
Epoch:   7 | T_Loss 0.214 | T_acc 0.938 | V_Loss 0.223 | V_acc. 0.934 | 14.606
Epoch:   8 | T_Loss 0.194 | T_acc 0.943 | V_Loss 0.205 | V_acc. 0.938 | 15.849
Epoch:   9 | T_Loss 0.178 | T_acc 0.947 | V_Loss 0.191 | V_acc. 0.941 | 13.956
Epoch:  10 | T_Loss 0.163 | T_acc 0.953 | V_Loss  0.18 | V_acc. 0.944 | 13.761

Training completed!
```

- 각 epoch에 따라 로그를 출력
  1) Train Loss
  2) Train Acc.
  3) Validation Loss
  4) Validation Acc.
  5) 학습 소요 시간

PyTorch Framework Flow

PIAI Research Department

- Deep Learning 모델 구축 과정



Model Testing

PIAI Research Department

- **딥러닝 모델 정확도 산정**

```
24 # 정확도 검증
25 with torch.no_grad():
26     test_loss, test_acc = epoch(test_loader)
27     test_acc = round(test_acc, 4)
28     test_loss = round(test_loss, 4)
29     print('Test Acc.: {}'.format(test_acc))
30     print('Test Loss: {}'.format(test_loss))
```

```
 Training completed!
Test Acc.: 0.952
Test Loss: 0.1617
```

You can install TorchMetrics using pip or conda:

```
# Python Package Index (PyPI)
pip install torchmetrics
# Conda
conda install -c conda-forge torchmetrics
```

- 학습 완료 후 Test데이터 10,000장을 모델에 넣고 분류 (torch.no_grad() = 역전파 학습 off)
- 모델이 분류한 결과와 Ground Truth값을 비교하여 모델의 정확도 (Acc.) 산출
- PyTorch의 torchmetrics 기능을 통하여 다양한 평가 Metric 사용 가능

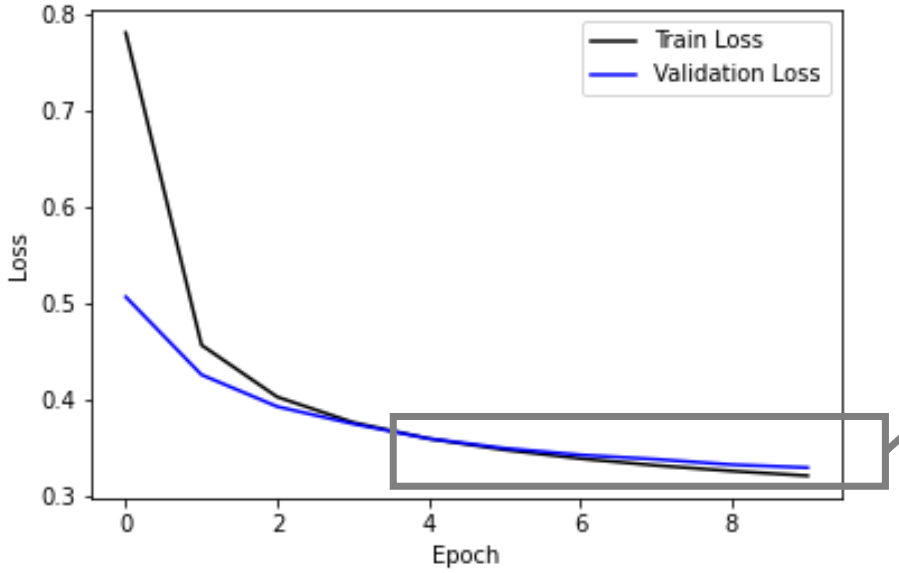
출처: https://torchmetrics.readthedocs.io/en/stable/pages/quickstart.html

Model Testing Visualization

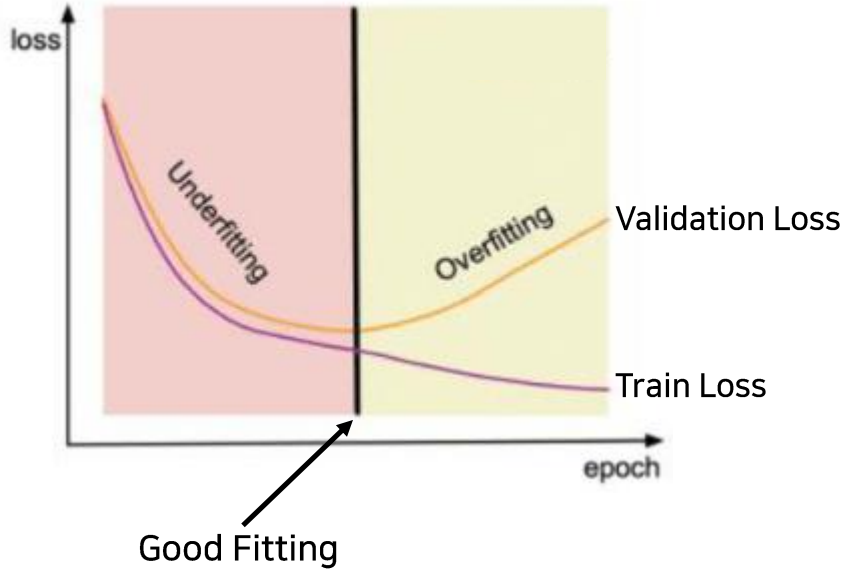
PIAI Research Department

- **모델의 학습 추이를 그래프로 확인**

```
plt.plot(range(0, maximum_epoch, 1), losses_train, label = 'Train Lo
plt.plot(range(0, maximum_epoch, 1), losses_val, label = 'Validation
plt.legend(loc = 'upper right')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```



- 학습 중 진행된 epoch에 따라 감소되는 Loss의 추이를 시각화
- Loss가 계속 줄어들고 있는 그래프 추이로 보아, 추가 학습 가능

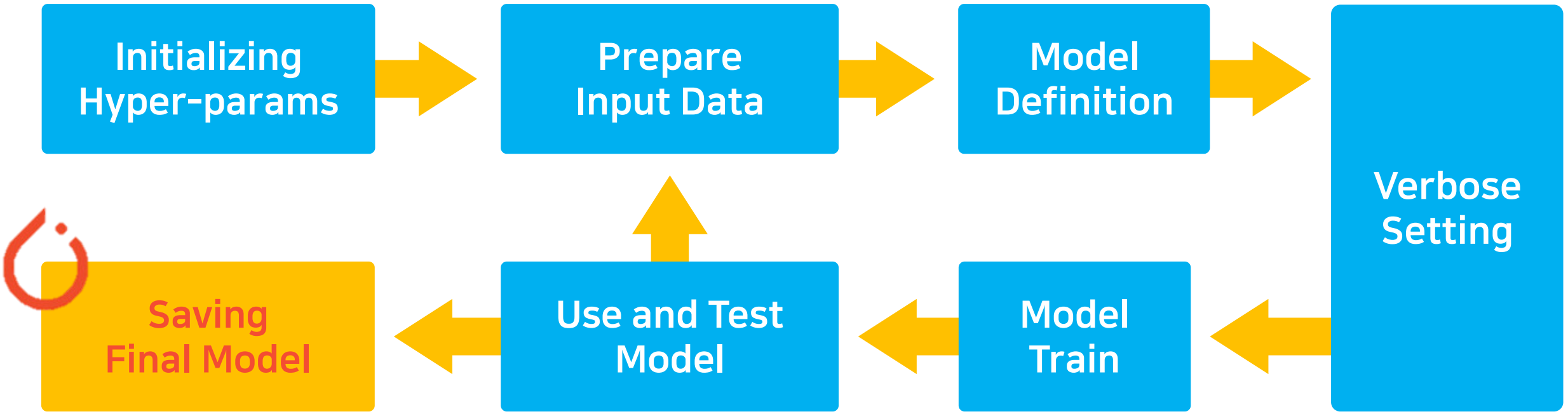


출처: https://www.researchgate.net/figure/a-Graphical-depiction-of-training-loss-with-varying-learning-rates-The-optimal_fig7_346206645

PyTorch Framework Flow

PIAI Research Department

- Deep Learning 모델 구축 과정



Model Saving/Loading

- **모델 저장 / 로드**

```
1  torch.save(net.state_dict(), './model.pth')
2
3  net = MLP(in_features = 784, out_features = 10).to(device)
4
5  net.load_state_dict(torch.load('./model.pth'))
```

<All keys matched successfully>

- 학습된 모델을 저장하여, 필요시 로드하여 사용 가능
- 저장할 경로를 지정하여 모델을 .pth 포맷으로 저장
- 저장한 모델을 로드하기 위하여, 비어 있는 모델을 선언(입 출력 개수)
- 선언한 모델에, 저장되어 있는 모델의 경로를 입력하여 로드 후 사용

Code Running

(student) 1_Pytorch Basic & Classifier.ipynb