ThunderSoft®

Edge Computing & On-Device AI Enabler

On-Device AI
지식 공유

2025.11

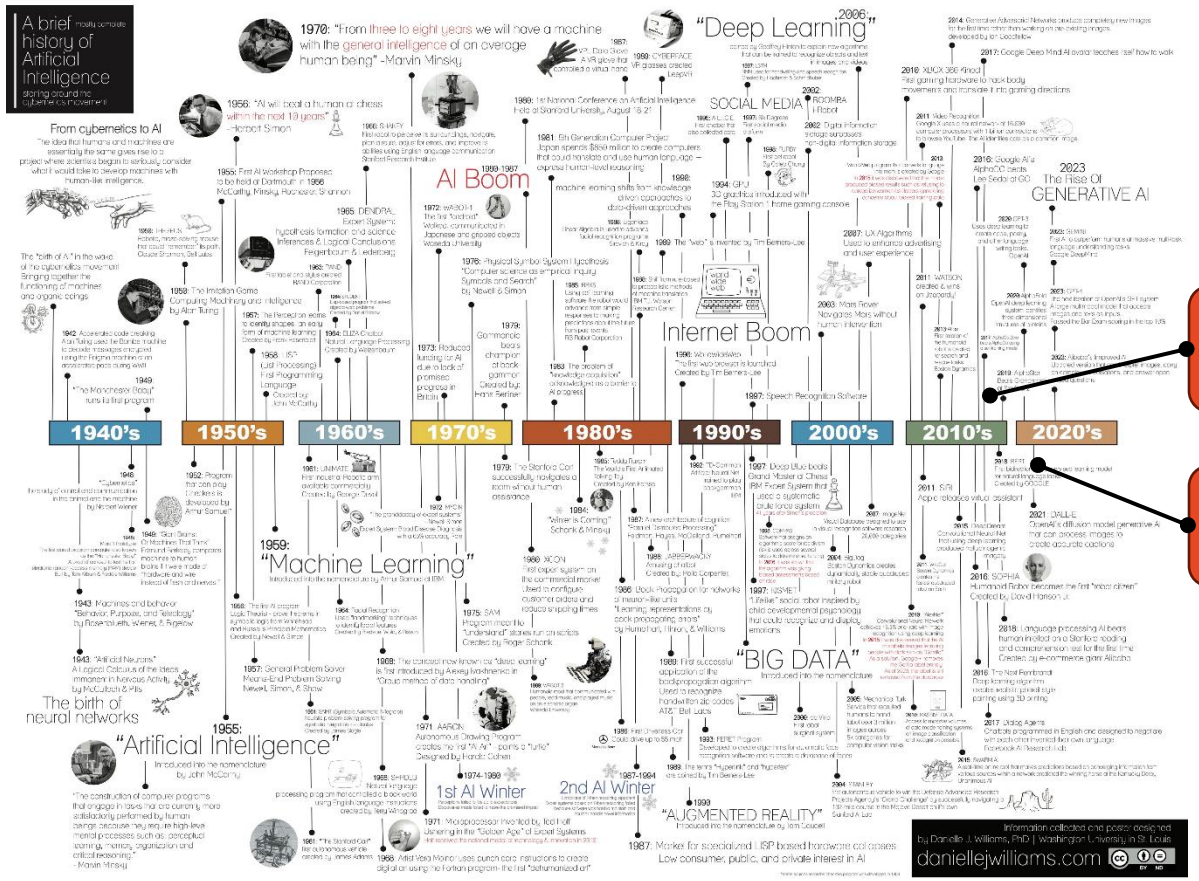
목차

용어 정리

- Perceptron : 여러 입력 신호를 받아 하나의 출력 신호를 만드는 가장 기본적인 인공 뉴런 모델.
- Batch : 학습할 때 한번에 처리하는 데이터 샘플의 묶음 단위.
- Iteration : 배치 하나를 사용해 모델이 한번 학습하는 과정을 의미.
- Epoch : 전체 학습 데이터를 한 번 모두 사용해 학습하는 한 사이클.
- Ground Truth : 모델이 예측할 때 비교하는 실제 정답 데이터.
- Loss Function : 모델 예측값과 실제값 간의 차이를 수치화해 평가하는 함수.
- Activation Function : 신경망에서 뉴런의 출력을 결정하며, 비선형성을 부여해 복잡한 학습이 가능하게 하는 함수.
- Backpropagation : 미분을 사용해 오차를 거꾸로 전파해 각 가중치를 조정하는 학습 알고리즘.
- Inference Driver : 추론 런타임의 API를 호출하여 추론을 수행하는 프로그램.
- Inference Runtime : DNN 추론을 실행하는 소프트웨어 환경.

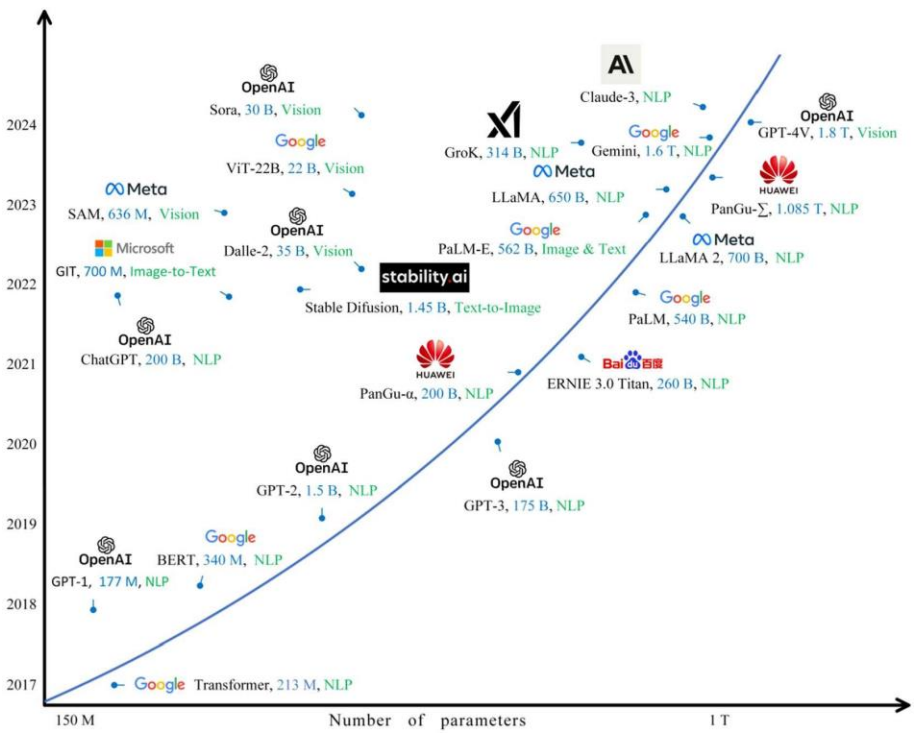
  LiteRT - Google, QNN - Qualcomm, TensorRT - Nvidia, NNC - Samsung
- LiteRT Interpreter : tflite 형식으로 컴파일된 모델을 실행해 추론을 수행하는 Interpreter.
- Delegate : 특정 하드웨어 가속기(GPU, NPU 등)를 활용할 수 있도록 중간에서 추론 작업을 넘기고 최적화하는 모듈. 이를 통해 성능과 에너지 효율을 높인다.

AI 역사



Attention (2017)

Chat GPT (2022)

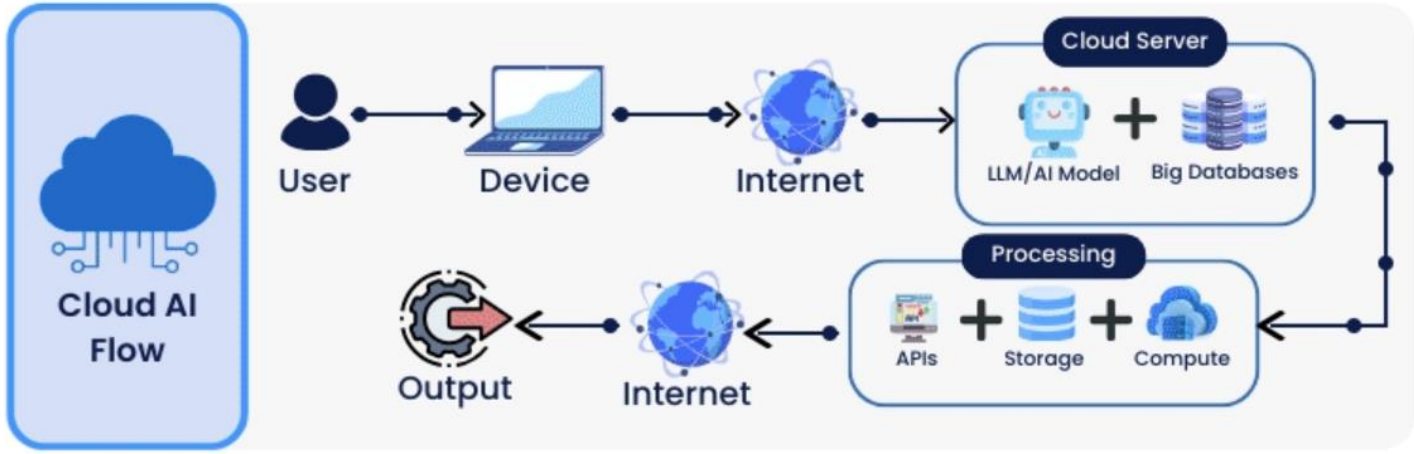


AI 역사

- **점점 거대해지는 모델들의 복잡한 연산을 감당하기 어려워 이를 보완하기 위해 클라우드 AI등장**

- **장점**
  - 강력한 컴퓨팅 파워 & 실시간 추론
  - 탄력적 확장성 & 부하 관리
  - 중앙집중

- **단점**
  - 네트워크 지연
  - 운영 비용
  - 데이터 보안
  - 인터넷 의존성



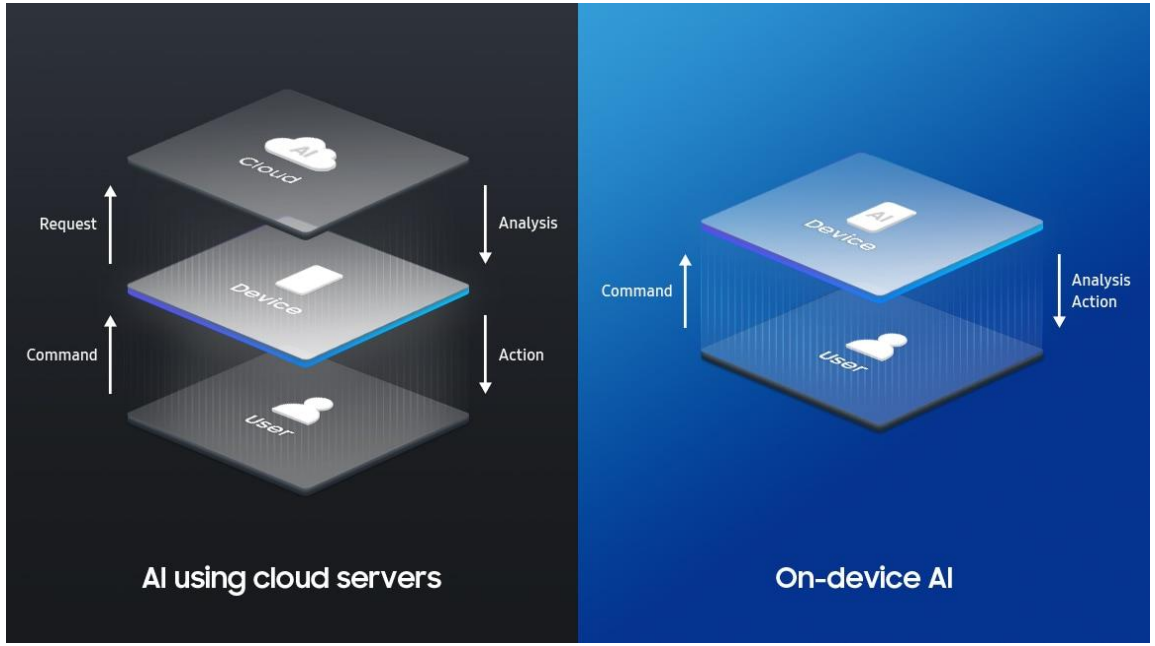
AI 역사 – On-Device AI의 필요성

❖ On-Device AI의 필요성

- 개인정보 보호 강화
- 실시간 처리와 빠른 응답
- 네트워크 의존도 감소
- 비용 절감
- 개인화 및 맞춤형 서비스

❖ On-Device AI의 한계와 도전 과제

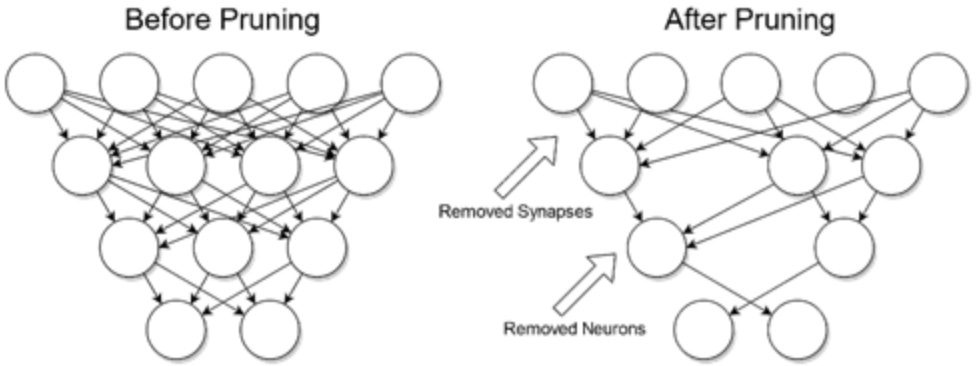
- 제한된 하드웨어 자원: 스마트폰이나 IoT 기기 등은 CPU, 메모리, 저장공간 등 자원이 제한적이어서 대규모 AI 모델을 온전히 실행하기 어려움.
- 모델 크기와 복잡성 제한: 대형 생성형 AI 모델 같은 복잡한 AI 연산 구현에 한계.
- 최적화와 에너지 효율: 연산 및 배터리 소비 최적화를 위한 하드웨어와 소프트웨어 개발이 필수적.
- 병렬 처리 및 연산 속도: 최신 AI 모델은 GPU 기반 대규모 병렬 처리를 요구하기 때문에, 동일한 연산을 모바일 칩셋에서 구현하는 데 기술적 난제가 존재.

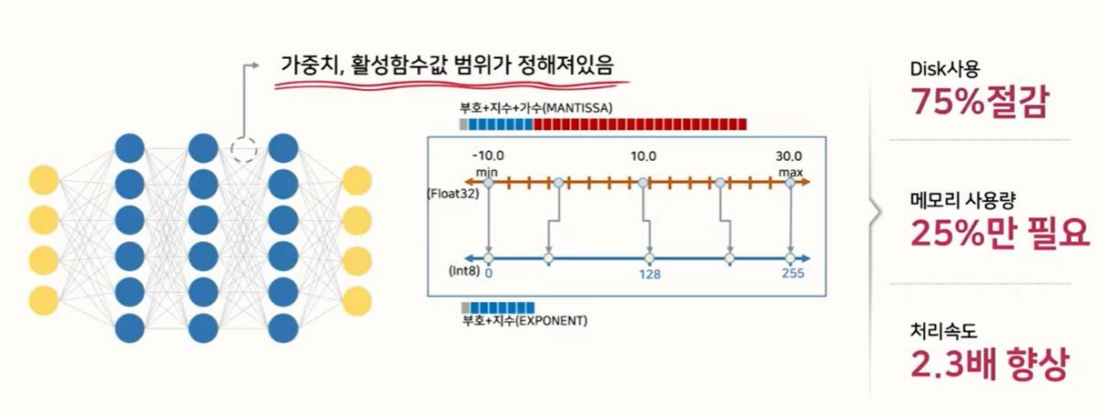


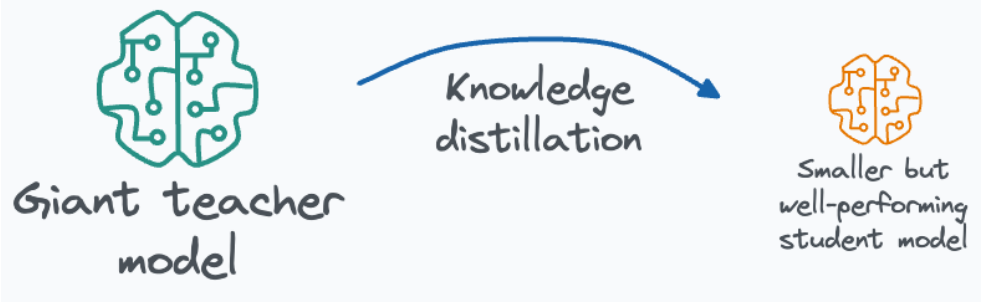
AI 역사 – On-Device AI의 필요성

❖ On-Device AI 모델 경량화 기법

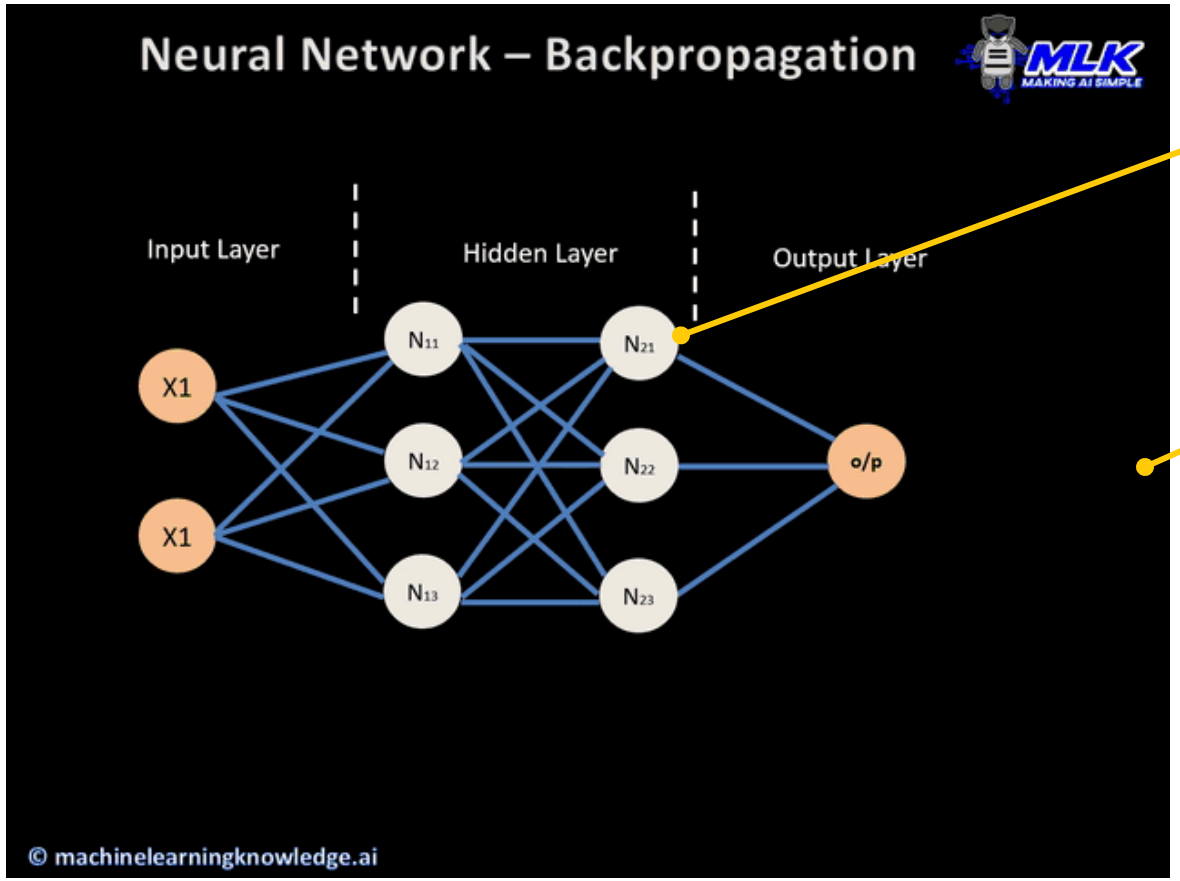
- Quantization : 모델의 가중치와 연산을 32비트에서 8비트 또는 그 이하로 줄여 메모리 사용량과 연산 부하를 크게 감소시킴
- Pruning : 중요하지 않은 뉴런이나 연결을 제거하여 모델 크기와 연산량을 줄임.
- Knowledge Distillation : 크고 복잡한 교사 모델이 작은 학생 모델에 지식을 전달하여 작은 모델도 높은 성능을 낼 수 있도록 학습시키는 기법
- Matrix/Tensor Decomposition : 모델 내부의 큰 행렬이나 텐서를 분해하여 연산량을 줄임.
- 하드웨어 가속기 활용과 플랫폼 최적화 : NPU, GPU등 하드웨어 가속기 적극 활용. 모바일 최적화 프레임워크를 통해 경량화 모델의 실행 효율을 극대화함.

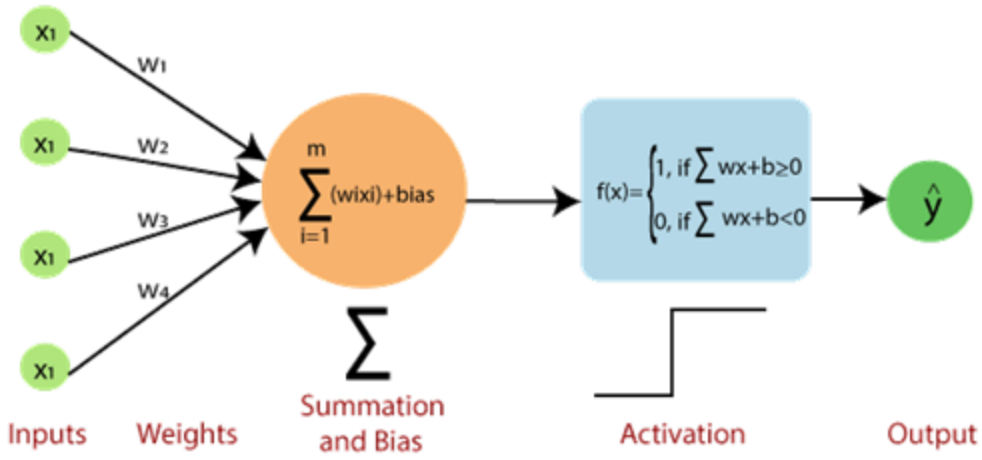






DNN 구조 및 과정 이해





LOSS FUNCTION | View-Point II : Maximum Likelihood

REVISIT DNN
11 / 17

Back to Machine Learning Problem

01. Collect training data

$$x = \{x_1, x_2, \dots, x_N\}$$
$$y = \{y_1, y_2, \dots, y_N\}$$
$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)\}$$

02. Define functions

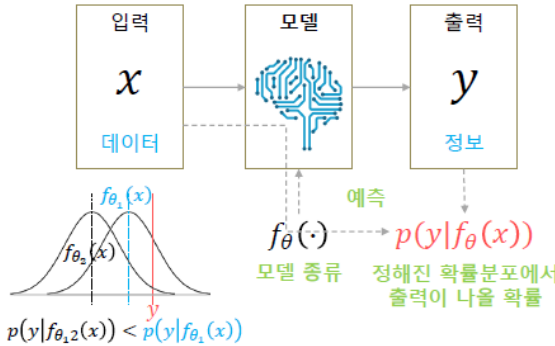
- Output : $f_\theta(x)$
- Loss : $-\log(p(y|f_\theta(x)))$

03. Learning/Training

Find the optimal parameter

04. Predicting/Testing

Compute optimal function output

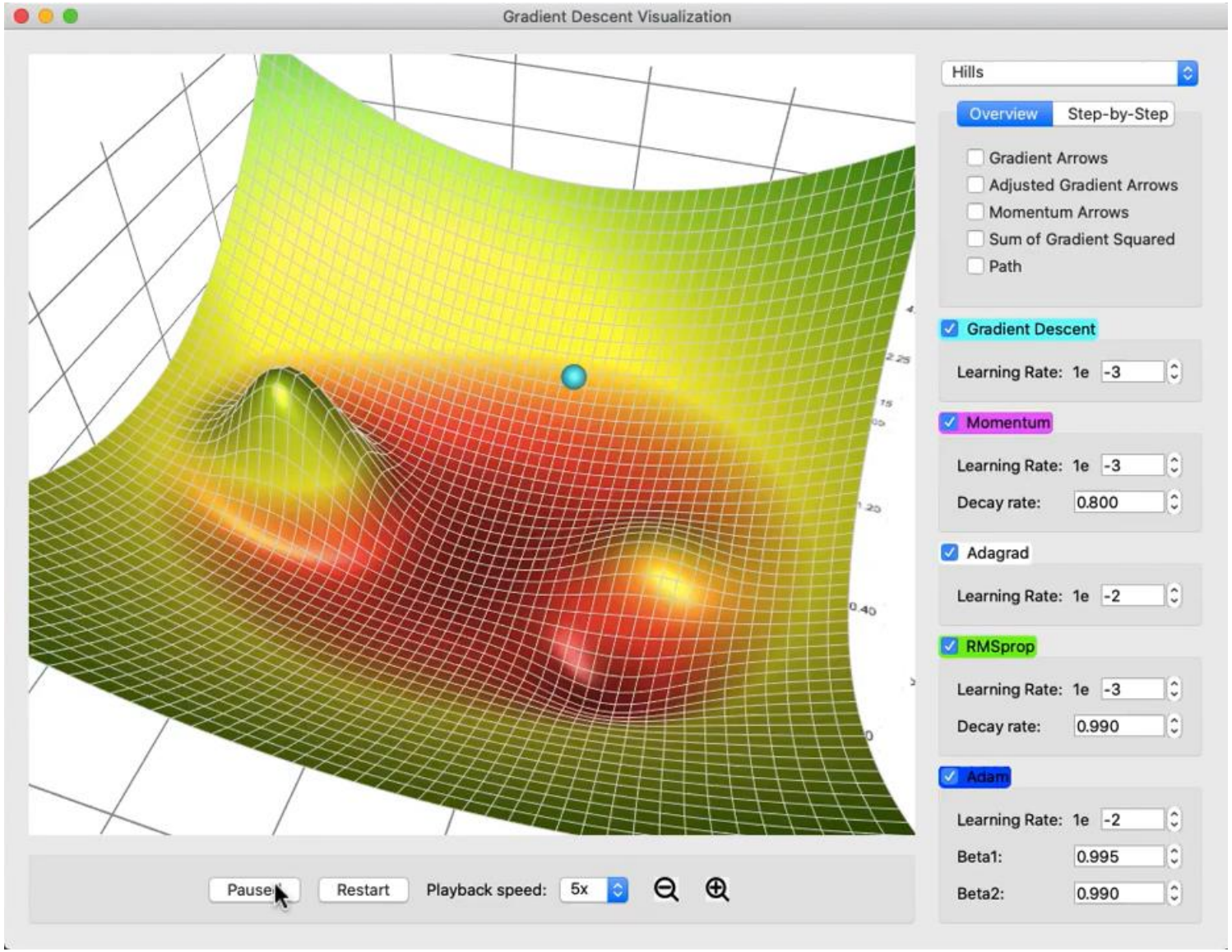


$$\theta^* = \underset{\theta}{\operatorname{argmin}}[-\log(p(y|f_\theta(x)))]$$ 주어진 데이터를 제일 잘 설명하는 모델 찾기

$$y_{new} \sim p(y|f_{\theta^*}(x_{new}))$$ 고정 입력, 고정/다른 출력

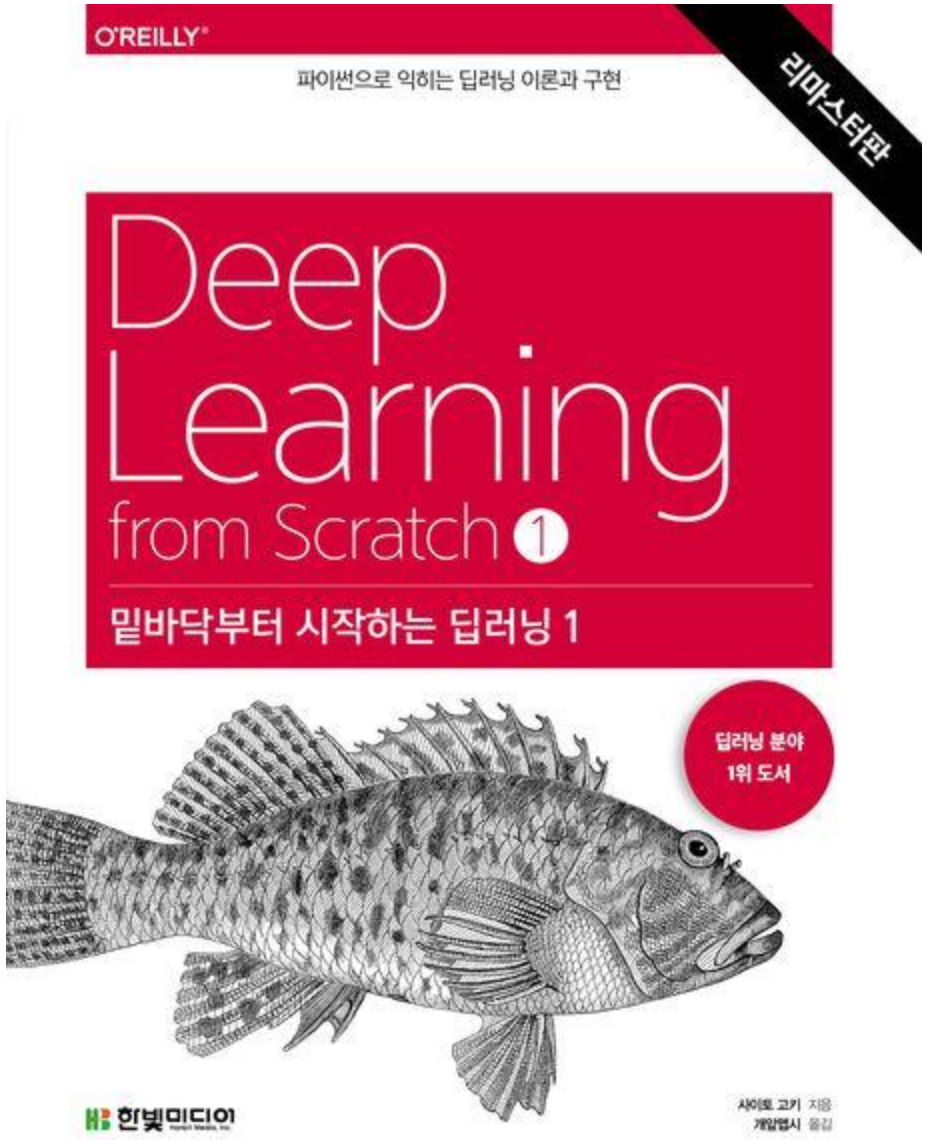
출처 : https://deepinsight.tistory.com/123

DNN 구조 및 과정 이해



출처 : https://medium.com/data-science/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

DNN 구조 및 과정 이해


O'REILLY
파이썬으로 익히는 딥러닝 이론과 구현
리마스터판
Deep Learning from Scratch 1
밑바닥부터 시작하는 딥러닝 1
딥러닝 분야 1위 도서
한빛미디어
사이토 고키 지음
개앞맵시 옮김

모델 만들기 – model.py

```python
import torch
import torch.nn as nn

# Define DNN for simple classifier
class SimpleClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(28*28, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 10)
        )

    def forward(self, x):
        return self.model(x)
```

모델 만들기 – train.py

```python
import os
import warnings

warnings.filterwarnings("ignore", category=DeprecationWarning)

import torch
import torch.nn as nn
import torch.optim as optim
from model import SimpleClassifier
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

transform = transforms.ToTensor()
train_data = datasets.MNIST(
    root="./data", train=True, download=True, transform=transform
)
train_loader = DataLoader(train_data, batch_size=64, shuffle=True)
test_data = datasets.MNIST(
    root="./data", train=False, download=True, transform=transform
)
test_loader = DataLoader(test_data, batch_size=64, shuffle=True)

# Create a model object
model = SimpleClassifier()

# Create objects for backpropagation and gradient descent
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=1e-3)
```

모델 만들기 – train.py

```python
# Train the model
epochs = 20
for epoch in range(epochs):
    for images, labels in train_loader:  # 60000 / 64 iterations in 1 epoch
        # Write code for training
        ## Hint: use ".view(images.size(0), -1)" to flatten the input images
        optimizer.zero_grad()
        pred = model(images.view(images.size(0), -1))
        loss = criterion(pred, labels)
        loss.backward()
        optimizer.step()

    if loss.item() < 0.5:
        break

    print(f"Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}")

# Evaluate the model with the test dataset
model.eval()
correct = 0
total = len(test_loader.dataset)

for images, labels in test_loader:
    pred = model(images.view(images.size(0), -1))

    # Postprocessing: hardmax is used for inference
    _, y = torch.max(pred, 1)

    # Accumulate the number of correct predictions
    correct += (y == labels).sum().item()

accuracy = 100 * correct / total
print(f"Accuracy: {accuracy:.2f}% " f"({correct}/{total} correct)")
```

모델 만들기 – train.py

```python
# Prepare a dummy input for ONNX export
sample_image, _ = train_data[0]
num_features = (
    sample_image.numel()
)  ## Hint: use ".numel()" to get the total number of features
dummy_input = torch.randn(1, num_features)

# Export to ONNX
# Call torch.onnx.export
torch.onnx.export(
    model,
    dummy_input,
    "./models/simple_classifier.onnx",
    input_names=["input"],
    output_names=["output"],
)

print("Successfully saved simple_classifier.onnx in ./models")
```

모델 만들기 – convert.py

```python
import argparse
from onnx2tf import convert

def convert_onnx_to_litert(onnx_path, output_dir):
    # Write code to call convert function
    convert(
        input_onnx_file_path=onnx_path,
        output_folder_path=output_dir,
        copy_onnx_input_output_names_to_tflite=True,
        non_verbose=False,
        not_use_onnxsim=True,
        not_use_opname_auto_generate=True,
    )

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Convert .onnx ONNX model to .tflite.")
    parser.add_argument(
        "--onnx-path",
        type=str,
        default="./models/simple_classifier.onnx",
        help="Path to the input .onnx model",
    )
    parser.add_argument(
        "--output-dir",
        type=str,
        default="./models/litert",
        help="Directory to save LiteRT model",
    )
    args = parser.parse_args()

    convert_onnx_to_litert(args.onnx_path, args.output_dir)
```

모델 만들기

```
(.venv) yechanyun@Yechanui-MacBookPro:OnDeviceAI-Bootcamp(main⚡) » python python/train.py
Epoch [1/20], Loss: 2.2521
Epoch [2/20], Loss: 2.2316
Epoch [3/20], Loss: 2.1042
Epoch [4/20], Loss: 2.0168
Epoch [5/20], Loss: 1.7330
Epoch [6/20], Loss: 1.5328
Epoch [7/20], Loss: 1.2190
Epoch [8/20], Loss: 0.9645
Epoch [9/20], Loss: 0.8111
Epoch [10/20], Loss: 0.5140
Epoch [11/20], Loss: 0.8207
Epoch [12/20], Loss: 0.5194
Accuracy: 84.64% (8464/10000 correct)
[torch.onnx] Obtain model graph for `SimpleClassifier([...]` with `torch.export.export(...,
strict=False)`...
[torch.onnx] Obtain model graph for `SimpleClassifier([...]` with `torch.export.export(...,
strict=False)`... ✅
[torch.onnx] Run decomposition...
[torch.onnx] Run decomposition... ✅
[torch.onnx] Translate the graph into ONNX...
[torch.onnx] Translate the graph into ONNX... ✅
Successfully saved simple_classifier.onnx in ./models
```

모델 만들기

```
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ python ./python/convert.py --onnx-path
models/simple_classifier.onnx --output-dir models/litert
Model loaded ====================================================================

Model conversion started ========================================================
INFO: input_op_name: input shape: [1, 784] dtype: float32
WARNING: The optimization process for shape estimation is skipped because it contains OPs that cannot
be inferred by the standard onnxruntime.
WARNING: name 'ort' is not defined

INFO: 2 / 6
INFO: onnx_op_type: Gemm onnx_op_name: node_linear
INFO:  input_name.1: input shape: [1, 784] dtype: float32
INFO:  input_name.2: model.0.weight shape: [128, 784] dtype: float32
INFO:  input_name.3: model.0.bias shape: [128] dtype: float32
INFO:  output_name.1: linear shape: [1, 128] dtype: float32
INFO: tf_op_type: matmul
INFO:  input.1.x: name: Placeholder:0 shape: (1, 784) dtype: <dtype: 'float32'>
INFO:  input.2.y: shape: (784, 128) dtype: <dtype: 'float32'>
INFO:  input.3.z: shape: (128,) dtype: <dtype: 'float32'>
INFO:  input.4.alpha: val: 1.0
INFO:  input.5.beta: val: 1.0
INFO:  output.1.output: name: tf.__operators__.add/AddV2:0 shape: (1, 128) dtype: <dtype: 'float32'>

INFO: 3 / 6
INFO: onnx_op_type: Relu onnx_op_name: node_relu
INFO:  input_name.1: linear shape: [1, 128] dtype: float32
INFO:  output_name.1: relu shape: [1, 128] dtype: float32
INFO: tf_op_type: relu
INFO:  input.1.features: name: tf.__operators__.add/AddV2:0 shape: (1, 128) dtype: <dtype: 'float32'>
INFO:  output.1.output: name: tf.nn.relu/Relu:0 shape: (1, 128) dtype: <dtype: 'float32'>

INFO: 4 / 6
INFO: onnx_op_type: Gemm onnx_op_name: node_linear_1
INFO:  input_name.1: relu shape: [1, 128] dtype: float32
INFO:  input_name.2: model.2.weight shape: [64, 128] dtype: float32
INFO:  input_name.3: model.2.bias shape: [64] dtype: float32
INFO:  output_name.1: linear_1 shape: [1, 64] dtype: float32
INFO: tf_op_type: matmul
INFO:  input.1.x: name: Placeholder:0 shape: (1, 128) dtype: <dtype: 'float32'>
INFO:  input.2.y: shape: (128, 64) dtype: <dtype: 'float32'>
INFO:  input.3.z: shape: (64,) dtype: <dtype: 'float32'>
INFO:  input.4.alpha: val: 1.0
INFO:  input.5.beta: val: 1.0
INFO:  output.1.output: name: tf.__operators__.add_1/AddV2:0 shape: (1, 64) dtype: <dtype: 'float32'>
```

모델 만들기

```
INFO: 5 / 6
INFO: onnx_op_type: Relu onnx_op_name: node_relu_1
INFO:  input_name.1: linear_1 shape: [1, 64] dtype: float32
INFO:  output_name.1: relu_1 shape: [1, 64] dtype: float32
INFO: tf_op_type: relu
INFO:  input.1.features: name: tf.__operators__.add_1/AddV2:0 shape: (1, 64) dtype: <dtype: 'float32'>
INFO:  output.1.output: name: tf.nn.relu_1/Relu:0 shape: (1, 64) dtype: <dtype: 'float32'>

INFO: 6 / 6
INFO: onnx_op_type: Gemm onnx_op_name: node_linear_2
INFO:  input_name.1: relu_1 shape: [1, 64] dtype: float32
INFO:  input_name.2: model.4.weight shape: [10, 64] dtype: float32
INFO:  input_name.3: model.4.bias shape: [10] dtype: float32
INFO:  output_name.1: output shape: [1, 10] dtype: float32
INFO: tf_op_type: matmul
INFO:  input.1.x: name: Placeholder:0 shape: (1, 64) dtype: <dtype: 'float32'>
INFO:  input.2.y: shape: (64, 10) dtype: <dtype: 'float32'>
INFO:  input.3.z: shape: (10,) dtype: <dtype: 'float32'>
INFO:  input.4.alpha: val: 1.0
INFO:  input.5.beta: val: 1.0
INFO:  output.1.output: name: tf.__operators__.add_2/AddV2:0 shape: (1, 10) dtype: <dtype: 'float32'>

saved_model output started ==========================================================
saved_model output complete!
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1763018650.014038 1239614 devices.cc:76] Number of eligible GPUs (core count >= 8, compute
capability >= 0.0): 0 (Note: TensorFlow was not compiled with CUDA or ROCm support)
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1763018650.133239 1239614 tf_tfl_flatbuffer_helpers.cc:392] Ignored output_format.
W0000 00:00:1763018650.133283 1239614 tf_tfl_flatbuffer_helpers.cc:395] Ignored
drop_control_dependency.
WARNING: If you want tflite input OP name and output OP name to match ONNX input and output names,
convert them after installing "flatc". Also, do not use symbols such as slashes in input/output OP
names. To install flatc, run the following command:
wget https://github.com/PINTO0309/onnx2tf/releases/download/1.16.31/flatc.tar.gz && tar -zxvf
flatc.tar.gz && sudo chmod +x flatc && sudo mv flatc /usr/bin/
Float32 tflite output complete!
I0000 00:00:1763018650.201256 1239614 devices.cc:76] Number of eligible GPUs (core count >= 8, compute
capability >= 0.0): 0 (Note: TensorFlow was not compiled with CUDA or ROCm support)
W0000 00:00:1763018650.246020 1239614 tf_tfl_flatbuffer_helpers.cc:392] Ignored output_format.
W0000 00:00:1763018650.246069 1239614 tf_tfl_flatbuffer_helpers.cc:395] Ignored
drop_control_dependency.
WARNING: If you want tflite input OP name and output OP name to match ONNX input and output names,
convert them after installing "flatc". Also, do not use symbols such as slashes in input/output OP
names. To install flatc, run the following command:
wget https://github.com/PINTO0309/onnx2tf/releases/download/1.16.31/flatc.tar.gz && tar -zxvf
flatc.tar.gz && sudo chmod +x flatc && sudo mv flatc /usr/bin/
Float16 tflite output complete!
```

모델 만들기

Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.

https://colab.research.google.com/drive/1pQZ7GA-dFEx1-qA-cwS4Hk26Je7Qm8qc?usp=sharing

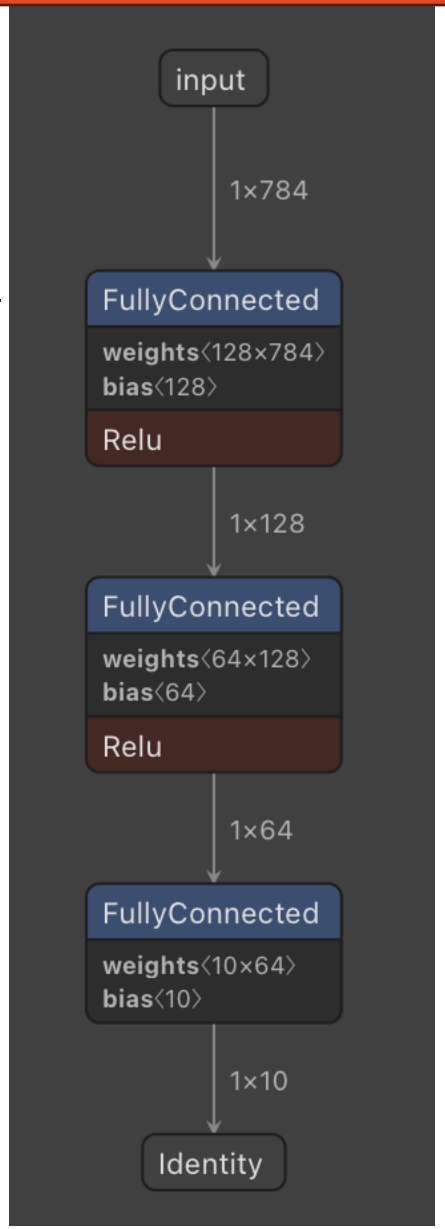
모델 만들기

Netron is a viewer for neural network, deep learning and machine learning models.
Netron supports ONNX, TensorFlow Lite, Core ML, Keras, Caffe, Darknet, PyTorch, TensorFlow.js…
Netron has experimental support for TorchScript, torch.export, ExecuTorch, TensorFlow, OpenVINO…

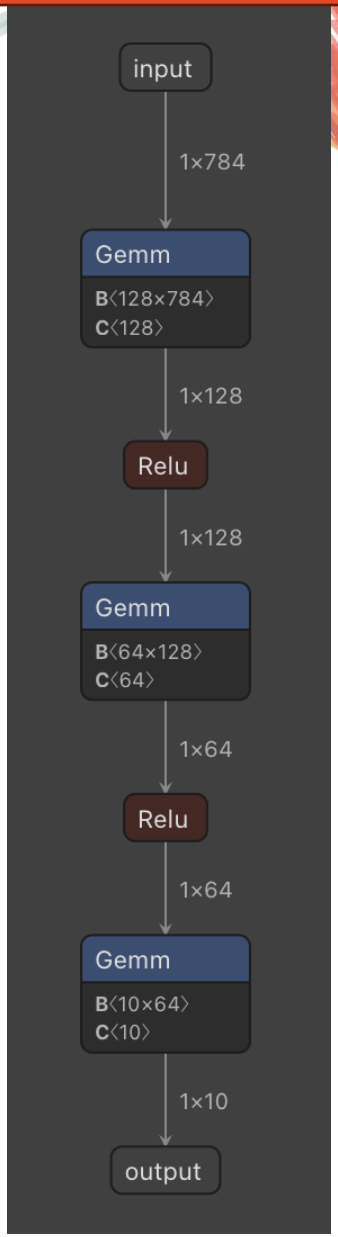
LUTZ ROEDER'S NETRON

https://netron.app/

simple_classifier_float32.tflite

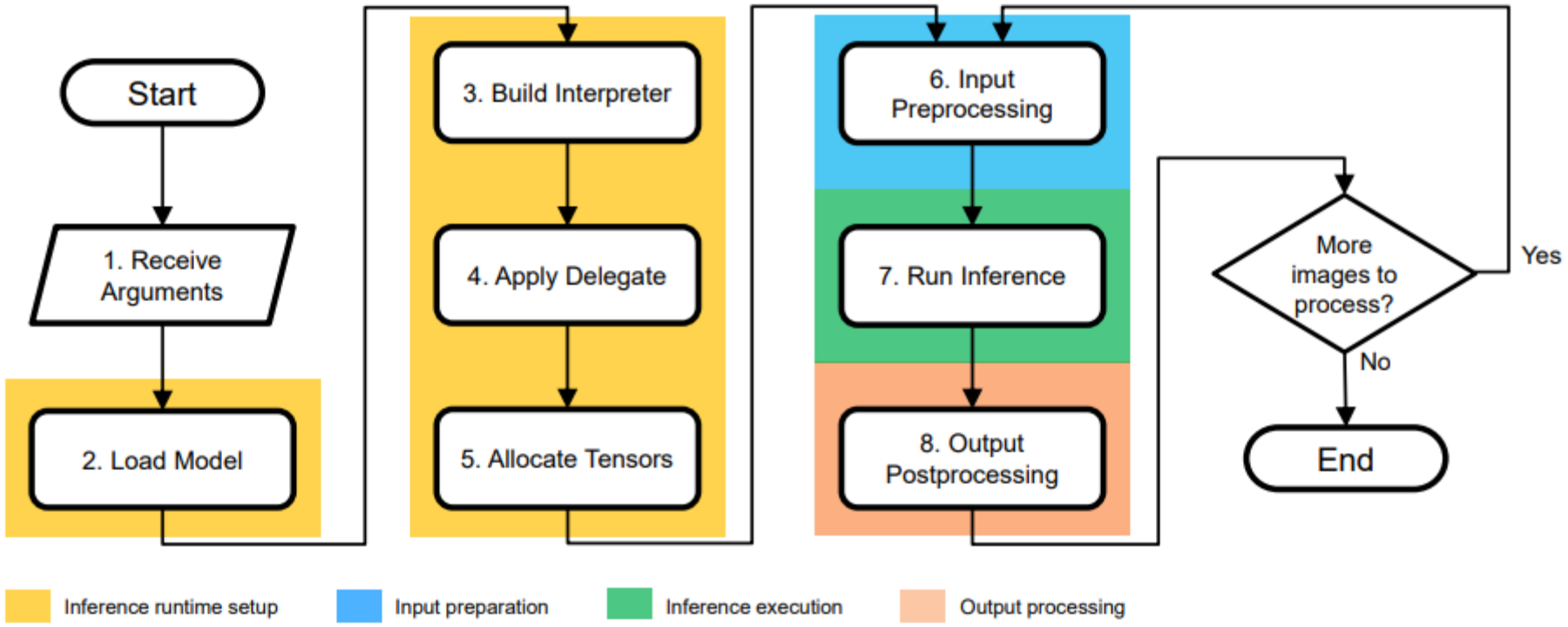


simple_classifier.onnx



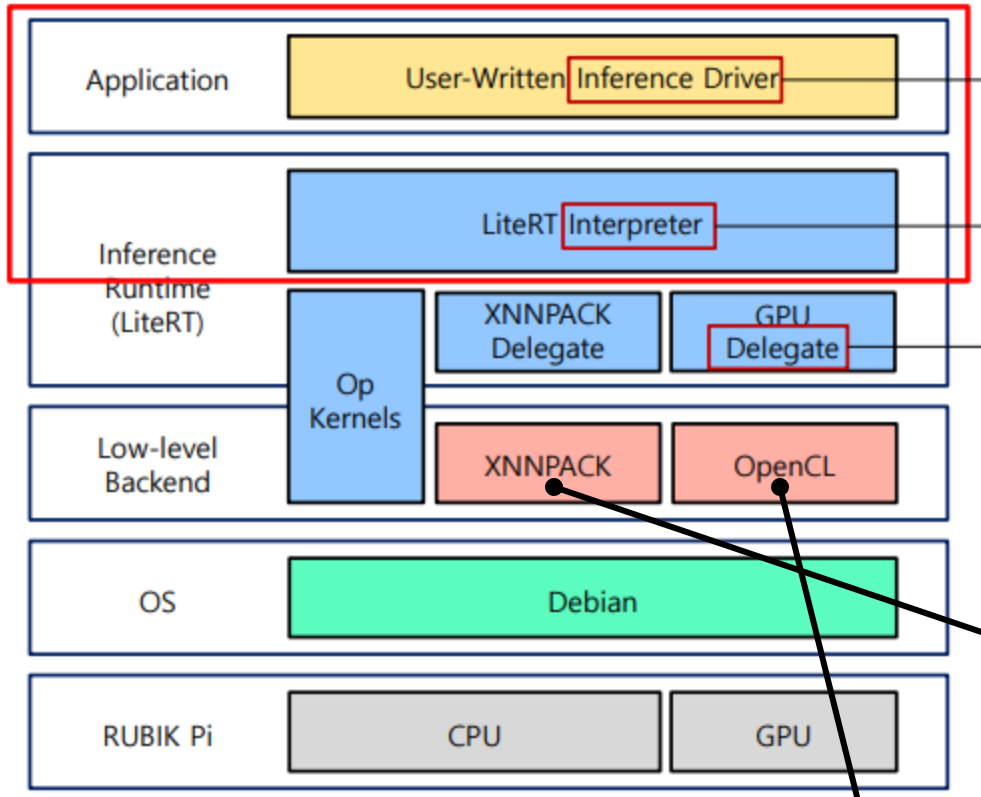
On-Device 추론 구현

Inference Driver Code Flow



On-Device 추론 구현

Software Stack on RUBIK Pi 3

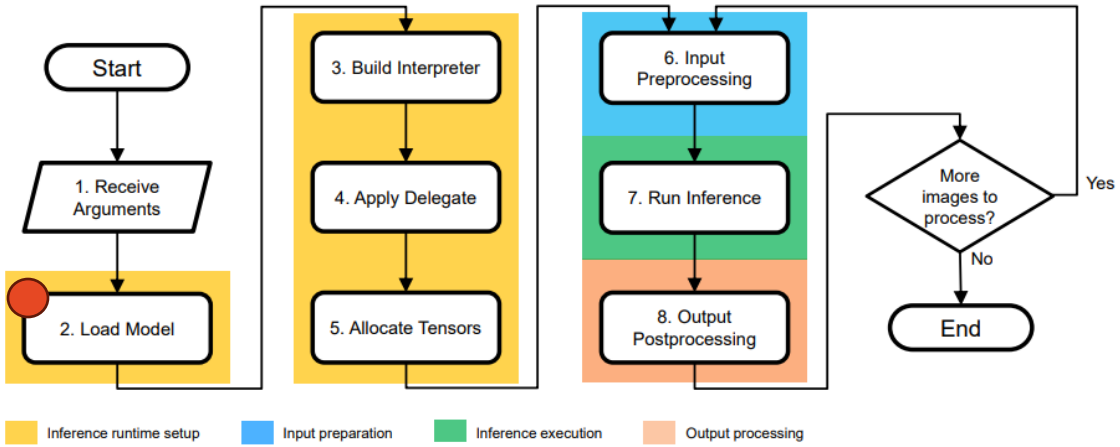


- Program that calls the inference runtime's APIs to drive inference
- A component that prepares and runs inference
- A component that offloads parts of a model to optimized implementations for faster or more efficient execution

XNNPACK is a highly optimized solution for neural network inference on ARM, x86, WebAssembly, and RISC-V platforms. It provides low-level performance primitives for accelerating high-level machine learning frameworks, such as TensorFlow Lite, TensorFlow.js, PyTorch, ONNX Runtime, ExecuTorch, and MediaPipe.

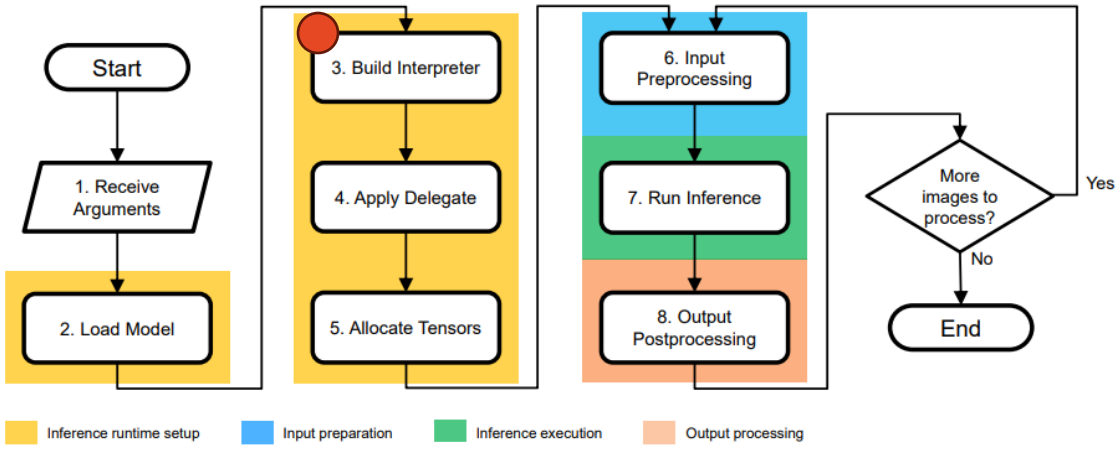
엔비디아 GPU 전용인 CUDA에 대응하기 위해 Apple이 2009년 8월에 발표한 GPGPU 프로그래밍 전용 API.

On-Device 추론 구현



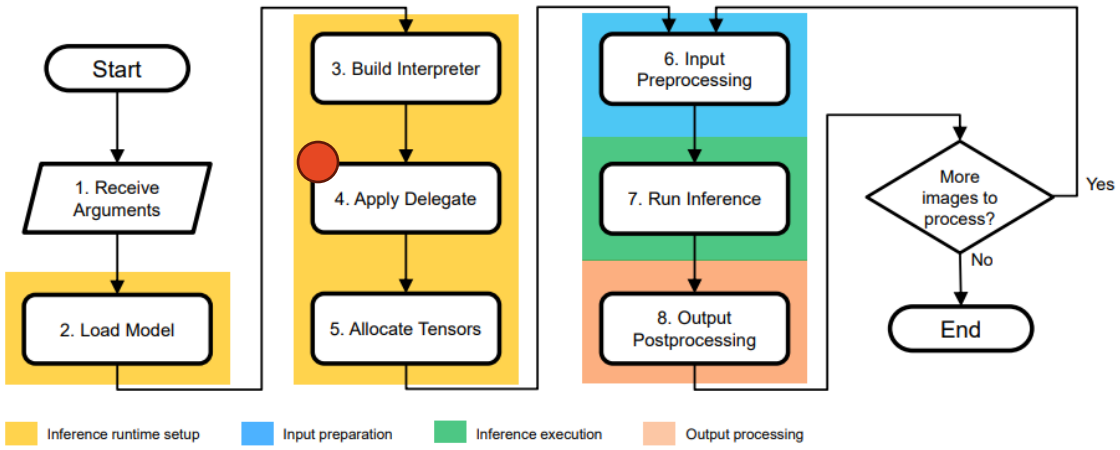

```
/* Load model */
std::unique_ptr<tflite::FlatBufferModel> model =
    tflite::FlatBufferModel::BuildFromFile(model_path.c_str());
```

On-Device 추론 구현



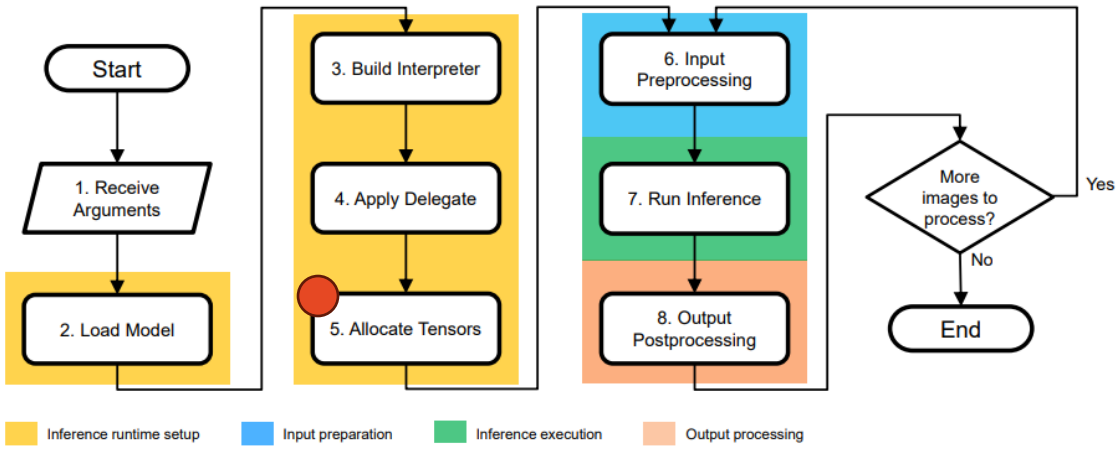

```cpp
/* Build interpreter */
// 1. Create an OpResolver
// 2. Create an interpreter builder
// 3. Build an interpreter using the interpreter builder
tflite::ops::builtin::BuiltinOpResolver resolver;
tflite::InterpreterBuilder builder(*model, resolver);
std::unique_ptr<tflite::Interpreter> interpreter;
builder(&interpreter);
if (!interpreter) {
    std::cerr << "Failed to initialize interpreter" << std::endl;
    return 1;
}
```

On-Device 추론 구현



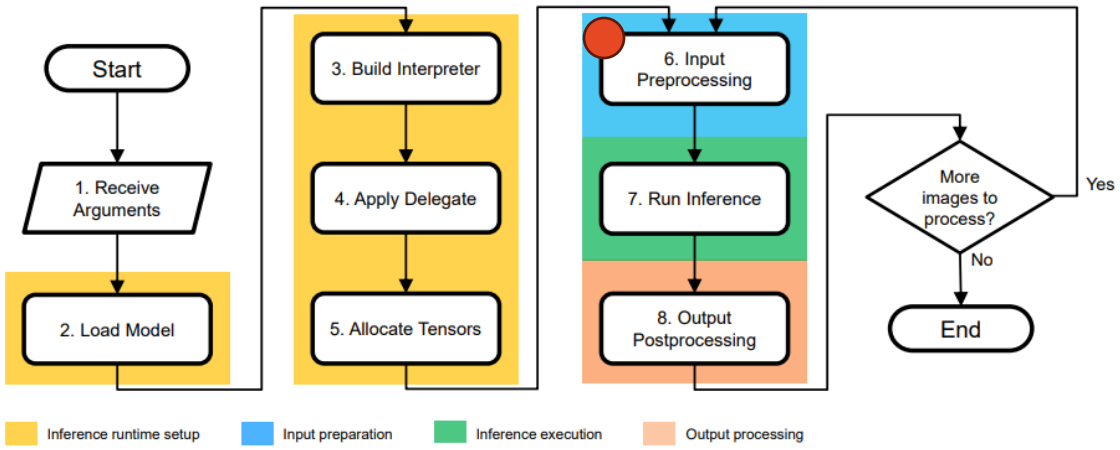

```cpp
/* Apply either XNNPACK delegate or GPU delegate */
// 1. Create a XNNPACK delegate
// 2. Create a GPU delegate
// 3. Apply either XNNPACK or GPU delegate
// 4. Delete the unsed delegate
TfLiteDelegate* xnn_delegate = TfLiteXNNPackDelegateCreate(nullptr);
TfLiteDelegate* gpu_delegate = TfLiteGpuDelegateV2Create(nullptr);

if (gpu_usage) {
    if (interpreter->ModifyGraphWithDelegate(gpu_delegate) == kTfLiteOk) {
        // Delete unused delegate
        if (xnn_delegate) {
            TfLiteXNNPackDelegateDelete(xnn_delegate);
        }
    } else {
        std::cerr << "Failed to Apply GPU Delegate" << std::endl;
    }
} else {
    if (interpreter->ModifyGraphWithDelegate(xnn_delegate) == kTfLiteOk) {
        // Delete unused delegate
        if (gpu_delegate) {
            TfLiteGpuDelegateV2Delete(gpu_delegate);
        }
    } else {
        std::cerr << "Failed to Apply XNNPACK Delegate" << std::endl;
    }
}
```

On-Device 추론 구현



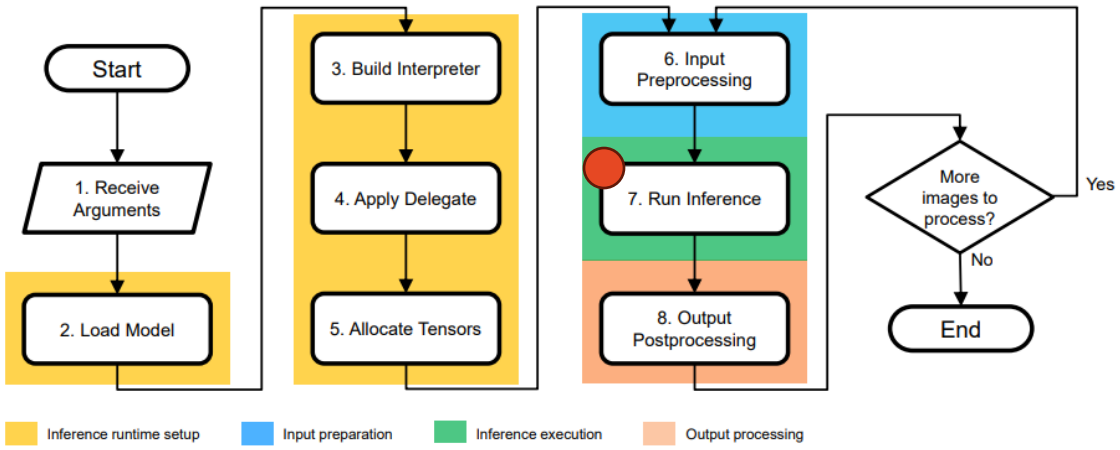

```
/* Allocate Tensors */
if (interpreter->AllocateTensors() != kTfLiteOk) {
    std::cerr << "Failed to Allocate Tensors" << std::endl;
    return 1;
}
```

On-Device 추론 구현



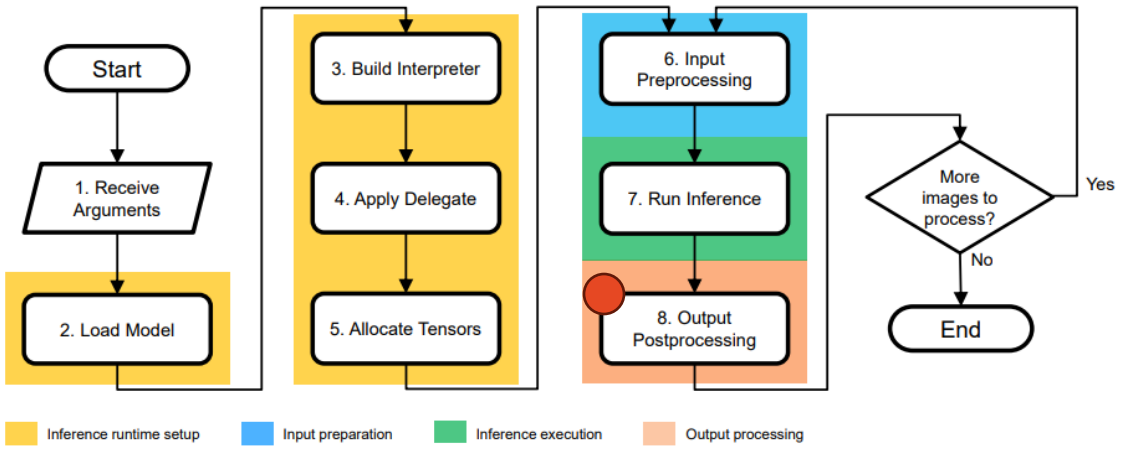

```cpp
// Preprocess input data
// 1. call util::preprocess_image to flatten a image
// 2. Copy the preprocessed data to input tensor
cv::Mat preprocessed_image = util::preprocess_image(image, 28, 28);

float* input_tensor = interpreter->typed_input_tensor<float>(0);
std::memcpy(input_tensor, preprocessed_image.ptr<float>(),
            preprocessed_image.total() * preprocessed_image.elemSize());
```

On-Device 추론 구현




```
/* Inference */
if (interpreter->Invoke() != kTfLiteOk) {
    std::cerr << "Failed to invoke the interpreter" << std::endl;
    return 1;
}
```

On-Device 추론 구현




```cpp
/* PostProcessing */
// 1. Get output tensor
// 2. Apply softmax to get probabilities
float *output_tensor = interpreter->typed_output_tensor<float>(0);
int num_classes = 10;
std::vector<float> logits(num_classes);
std::memcpy(logits.data(), output_tensor, sizeof(float) * num_classes);

std::vector<float> probs(num_classes);
util::softmax(logits.data(), probs, num_classes);


// ====================================
// Print Top-3 predictions every 10 iterations
if ((count + 1) % 10 == 0) {
    std::cout << "\n[INFO] Top 3 predictions for image index " << count << ":"
    << std::endl;
    auto top_k_indices = util::get_topK_indices(probs, 3);
    for (int idx : top_k_indices) {
        std::string label =
            class_labels_map.count(idx) ? class_labels_map[idx] : "unknown";
        std::cout << "- Class " << idx << " (" << label << "): "
            << probs[idx] << std::endl;
    }
}
```

On-Device 추론 구현

```
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ make inference
g++ -std=c++17 -O3 -w -Iinc -Isrc -I/usr/include -I/usr/include/opencv4 -
I/home/thundersoft/project/OnDeviceAI-Bootcamp/external/litert -c src/inference_driver.cpp -o
obj/inference_driver.o
g++ -std=c++17 -O3 -w -Iinc -Isrc -I/usr/include -I/usr/include/opencv4 -
I/home/thundersoft/project/OnDeviceAI-Bootcamp/external/litert -c src/util.cpp -o obj/util.o
g++ -std=c++17 -O3 -w -Llib obj/inference_driver.o obj/util.o -o bin/inference_driver -Wl,-
rpath=\$ORIGIN/../lib -ltensorflowlite -ltensorflowlite_gpu_delegate -lEGL -lGLESv2 -lopencv_core -
lopencv_imgproc -lopencv_imgcodecs -ljsoncpp
Inference driver built successfully.
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ ls bin/
inference_driver
```

On-Device 추론 구현

```bash
#!/bin/bash

set -e  # Exit if any command fails

# --------- CONFIGURATION ---------
executable="./bin/inference_driver"
model="./models/litert/simple_classifier_float32.tflite"
gpu_usage="true"
class_labels="class_labels.json"
image_dir="./data/MNIST/test"
total_inputs=1000
# ---------------------------------

# Sanity check for files and directories
if [ ! -f "$model" ]; then
    echo "ERROR: Model file not found: $model"
    exit 1
fi

if [ ! -d "$image_dir" ]; then
    echo "ERROR: Image directory not found: $image_dir"
    exit 1
fi

# Build array of image paths
image_paths=()
echo "Generating array of the first $total_inputs image paths..."
for ((i=0; i<total_inputs; i++)); do

    # Format the filename with leading zeros
    filename=$(printf "%05d.png" $i)
    image_path="$image_dir/$filename"

    # Check if the image file exists
    if [ ! -f "$image_path" ]; then
        echo "ERROR: Image file not found: $image_path"
        echo "Please check if 'total_inputs' exceeds the number of available images."
        exit 1
    fi
    image_paths+=("$image_path")
done

# Run
echo "Starting inference..."
"$executable" "$model" "$gpu_usage" "$class_labels" "${image_paths[@]}"
echo "Inference finished."
```

On-Device 추론 구현

```
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ ./run_inference_driver.sh
Generating array of the first 1000 image paths...
Starting inference...
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite delegate for GPU.
INFO: Loaded OpenCL library with dlopen.
W/Adreno-GSL (1214790,1214790): <os_lib_map:1488>:   os_lib_map error: libadreno_app_profiles.so:
cannot open shared object file: No such file or directory, on 'libadreno_app_profiles.so'

W/Adreno-CB (1214790,1214790): <cl_app_profiles_initialize:104>: Failed to load the app profiles
library libadreno_app_profiles.so!
INFO: Initialized OpenCL-based API.
INFO: Created 1 GPU delegate kernels.

[INFO] Top 3 predictions for image index 9:
- Class 9 (nine): 0.621158
- Class 7 (seven): 0.277299
- Class 4 (four): 0.0960337

[INFO] Top 3 predictions for image index 19:
- Class 4 (four): 0.712362
- Class 9 (nine): 0.242381
- Class 6 (six): 0.0130831

...

[INFO] Top 3 predictions for image index 999:
- Class 9 (nine): 0.458141
- Class 7 (seven): 0.281691
- Class 4 (four): 0.215371

[INFO] Average E2E latency (1000 runs): 0.459 ms

[INFO] Average Preprocessing latency (1000 runs): 0.015 ms

[INFO] Average Inference latency (1000 runs): 0.317 ms

[INFO] Average Postprocessing latency (1000 runs): 0 ms

[INFO] Throughput: 991.08 items/sec (1000 items in 1009 ms)
Inference finished.
```

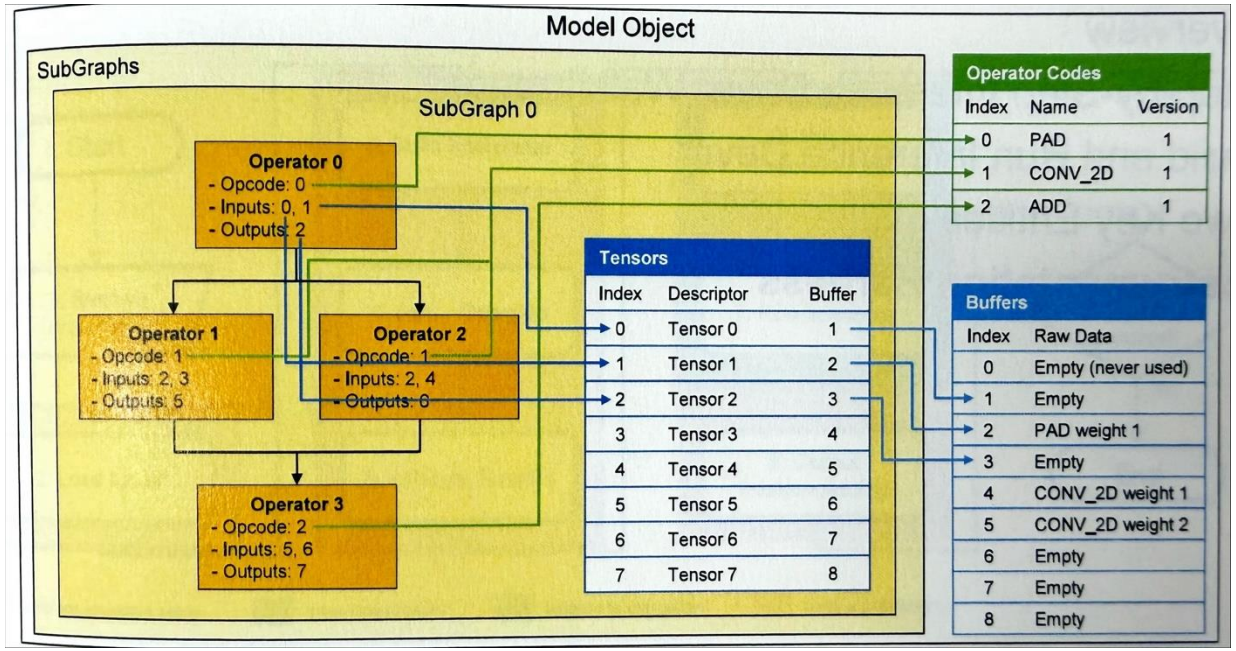
On-Device 추론 구현

❖ Two Key Entities of LiteRT

- Model Object
  - Static description of the model stored in a .tflite file.
- Interpreter
  - Runtime instance built from the model object.

❖ Model Object

- Subgraph
  - Computation graph with its own operators and tensors
- Operators
  - Basic unit of neural operations
  - Each operator references an operator code and tensors it uses
- Operator Code
  - Identifier that defines the type of neural operation
- Tensor
  - Descriptor of a multi-dimensional array instantiate at runtime
    - Name, shape, data type, buffer reference
- Buffer
  - Container of raw constant data referenced by immutable tensors
    - Weights and biases
    - Empty for mutable tensors

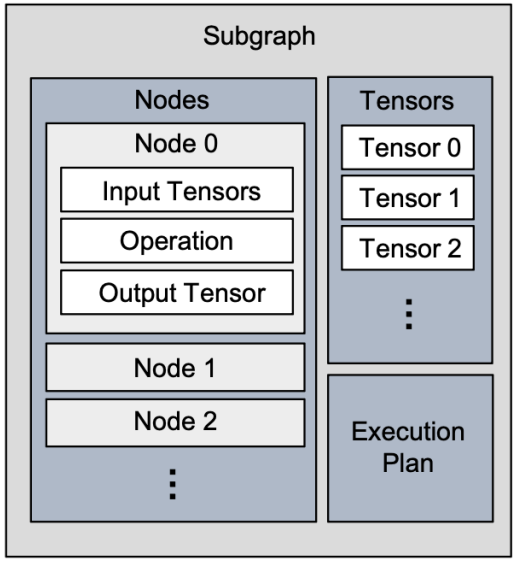


https://www.tensorflow.org/mlir/tfl_ops

On-Device 추론 구현

Internal Data Structures of LiteRT Model

1. Subgraph
   - An independent computation graph with its own nodes, tensors, and execution plan
     - Multiple subgraphs are used to support advanced model features such as control flow and modular function calls
2. Tensor
   - Multi-dimensional array that holds inputs, outputs, weights and/or intermediate results
3. Node
   - Basic unit of neural operations
4. Execution plan
   - Ordered list of node indices
   - Defines execution sequence



On-Device 추론 구현

```
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ make harness
g++ -std=c++17 -O3 -w -Iinc -Isrc -I/usr/include -I/usr/include/opencv4 -
I/home/thundersoft/project/OnDeviceAI-Bootcamp/external/litert -c src/instrumentation_harness.cpp -o
obj/instrumentation_harness.o
g++ -std=c++17 -O3 -w -Iinc -Isrc -I/usr/include -I/usr/include/opencv4 -
I/home/thundersoft/project/OnDeviceAI-Bootcamp/external/litert -c src/instrumentation_harness_utils.cpp
-o obj/instrumentation_harness_utils.o
g++ -std=c++17 -O3 -w -Llib obj/instrumentation_harness.o obj/instrumentation_harness_utils.o -o
bin/instrumentation_harness -Wl,-rpath=\$ORIGIN/../lib -ltensorflowlite -ltensorflowlite_gpu_delegate -
lEGL -lGLESv2 -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -ljsoncpp
Instrumentation harness built successfully.
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ ./bin/instrumentation_harness
Usage: ./bin/instrumentation_harness<model_path> [gpu_usage]
(.venv) thundersoft@RUBIKPi:~/project/OnDeviceAI-Bootcamp$ ./bin/instrumentation_harness
./models/litert/simple_classifier_float32.tflite
cat /proc/1244656/maps | grep tflite
7faabd4000-7faac40000 r--s 00000000 08:05 1484292
 /home/thundersoft/project/OnDeviceAI-Bootcamp/models/litert/simple_classifier_float32.tflite
7fb6720000-7fb6b45000 r-xp 00000000 08:05 1842252
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/delegates/gpu/libtensorflowlite_gpu_delegate.so
7fb6b45000-7fb6b5e000 ---p 00425000 08:05 1842252
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/delegates/gpu/libtensorflowlite_gpu_delegate.so
7fb6b5e000-7fb6b70000 r--p 0042e000 08:05 1842252
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/delegates/gpu/libtensorflowlite_gpu_delegate.so
7fb6b70000-7fb6b71000 rw-p 00440000 08:05 1842252
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/delegates/gpu/libtensorflowlite_gpu_delegate.so
7fb6b80000-7fb70c6000 r-xp 00000000 08:05 1222157
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/libtensorflowlite.so
7fb70c6000-7fb70d9000 ---p 00546000 08:05 1222157
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/libtensorflowlite.so
7fb70d9000-7fb70e0000 r--p 00549000 08:05 1222157
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/libtensorflowlite.so
7fb70e0000-7fb70e7000 rw-p 00550000 08:05 1222157
 /home/thundersoft/.cache/bazel/_bazel_thundersoft/b335a94f3f6d40114915f8ee7e51949c/execroot/litert/baz
el-out/aarch64-opt/bin/tflite/libtensorflowlite.so
Press Enter to continue...
```

On-Device 추론 구현

```
Schema version of the model: 3
Supported schema version: 3
Press Enter to continue...

Total 1 operator codes in the model
[0] FULLY_CONNECTED, version: 1, supported (Y/N): Y
Press Enter to continue...

Number of subgraphs: 1
Press Enter to continue...
SubGraph [0] main
Total 10 tensors in SubGraph [0]
Tensor [0] input, type = FLOAT32, shape = [1, 784], buffer = 1 (empty)
Tensor [1] arith.constant, type = FLOAT32, shape = [10, 64], buffer = 2 (has data, size = 2560)
Tensor [2] arith.constant1, type = FLOAT32, shape = [64, 128], buffer = 3 (has data, size = 32768)
Tensor [3] arith.constant2, type = FLOAT32, shape = [64], buffer = 4 (has data, size = 256)
Tensor [4] arith.constant3, type = FLOAT32, shape = [128], buffer = 5 (has data, size = 512)
Tensor [5] arith.constant4, type = FLOAT32, shape = [10], buffer = 6 (has data, size = 40)
Tensor [6] model/tf.linalg.matmul/MatMul, type = FLOAT32, shape = [128, 784], buffer = 7 (has data,
size = 401408)
Tensor [7] model/tf.linalg.matmul/MatMul;model/tf.nn.relu/Relu;model/tf.__operators__.add/AddV2, type =
FLOAT32, shape = [1, 128], buffer = 8 (empty)
Tensor [8] model/tf.linalg.matmul_1/MatMul;model/tf.nn.relu_1/Relu;model/tf.__operators__.add_1/AddV2,
type = FLOAT32, shape = [1, 64], buffer = 9 (empty)
Tensor [9] Identity, type = FLOAT32, shape = [1, 10], buffer = 10 (empty)
Press Enter to continue...
```

On-Device 추론 구현

```
Total 3 nodes in SubGraph [0]
Node [0]: FULLY_CONNECTED
  Input tensors: 0 6 4
  Output tensors: 7
Node [1]: FULLY_CONNECTED
  Input tensors: 7 2 3
  Output tensors: 8
Node [2]: FULLY_CONNECTED
  Input tensors: 8 1 5
  Output tensors: 9
Press Enter to continue...

Number of subgraphs: 1
Number of nodes of subgraph 0: 3
Number of tensors in subgraph 0: 10
Execution plan size of subgraph 0: 3
Node 0: FULLY_CONNECTED
Node 1: FULLY_CONNECTED
Node 2: FULLY_CONNECTED
Press Enter to continue...
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
INFO: Created TensorFlow Lite delegate for GPU.

Number of nodes of subgraph 0: 4
Node 0: FULLY_CONNECTED
Node 1: FULLY_CONNECTED
Node 2: FULLY_CONNECTED
Node 3: DELEGATE
Press Enter to continue...
```

On-Device 추론 구현

```
Execution plan size of subgraph 0: 1
Node 3: DELEGATE
Press Enter to continue...

Inputs:
0 (type: FLOAT32, dims: [1, 784])
1 (type: FLOAT32, dims: [10, 64])
2 (type: FLOAT32, dims: [64, 128])
3 (type: FLOAT32, dims: [64])
4 (type: FLOAT32, dims: [128])
5 (type: FLOAT32, dims: [10])
6 (type: FLOAT32, dims: [128, 784])

Outputs:
9 (type: FLOAT32, dims: [1, 10])

Total 8 tensors are used.
Press Enter to continue...

==== Before Allocate Tensors ====
Tensor 0: input | AllocType: kTfLiteArenaRw | Shape: [1, 784] | Bytes: 3136 | Address: 0
Tensor 1: arith.constant | AllocType: kTfLiteMmapRo | Shape: [10, 64] | Bytes: 2560 | Address:
0x7faac3e4e8
Tensor 2: arith.constant1 | AllocType: kTfLiteMmapRo | Shape: [64, 128] | Bytes: 32768 | Address:
0x7faac364d4
Tensor 3: arith.constant2 | AllocType: kTfLiteMmapRo | Shape: [64] | Bytes: 256 | Address: 0x7faac363c8
Tensor 4: arith.constant3 | AllocType: kTfLiteMmapRo | Shape: [128] | Bytes: 512 | Address:
0x7faac361bc
Tensor 5: arith.constant4 | AllocType: kTfLiteMmapRo | Shape: [10] | Bytes: 40 | Address: 0x7faac36188
Tensor 6: model/tf.linalg.matmul/MatMul | AllocType: kTfLiteMmapRo | Shape: [128, 784] | Bytes: 401408
| Address: 0x7faabd417c
Tensor 9: Identity | AllocType: kTfLiteArenaRw | Shape: [1, 10] | Bytes: 40 | Address: 0
Press Enter to continue...
```

On-Device 추론 구현

```
==== After Allocate Tensors ====
Tensor 0: input | AllocType: kTfLiteArenaRw | Shape: [1, 784] | Bytes: 3136 | Address: 0x5586b40b00
Tensor 1: arith.constant | AllocType: kTfLiteMmapRo | Shape: [10, 64] | Bytes: 2560 | Address:
0x7faac3e4e8
Tensor 2: arith.constant1 | AllocType: kTfLiteMmapRo | Shape: [64, 128] | Bytes: 32768 | Address:
0x7faac364d4
Tensor 3: arith.constant2 | AllocType: kTfLiteMmapRo | Shape: [64] | Bytes: 256 | Address: 0x7faac363c8
Tensor 4: arith.constant3 | AllocType: kTfLiteMmapRo | Shape: [128] | Bytes: 512 | Address:
0x7faac361bc
Tensor 5: arith.constant4 | AllocType: kTfLiteMmapRo | Shape: [10] | Bytes: 40 | Address: 0x7faac36188
Tensor 6: model/tf.linalg.matmul/MatMul | AllocType: kTfLiteMmapRo | Shape: [128, 784] | Bytes: 401408
| Address: 0x7faabd417c
Tensor 9: Identity | AllocType: kTfLiteArenaRw | Shape: [1, 10] | Bytes: 40 | Address: 0x5586b41740
Press Enter to continue...
0: Node 3 -> TfLiteXNNPackDelegate
Press Enter to continue...
```

On-Device 추론 구현

On-Device 추론 구현

ThunderSoft

DEMO

Q & A