

On-device AI 용 Neural Network Trainer 개발

- Pave the way to On-device Multi-modal LLMs -

창의적통합설계1 스펙 발표

2024.9.20

Team A

송우경

장민혁

이현우

Contents

- Overview
 - What is NNTrainer?
- Goal/Problem & Requirement & Approach
- Development Environment
- Architecture
- Implementation Spec
- Detailed Plan
- Demo Plan
- Division and Assignment of Work
- Schedule

On-device AI

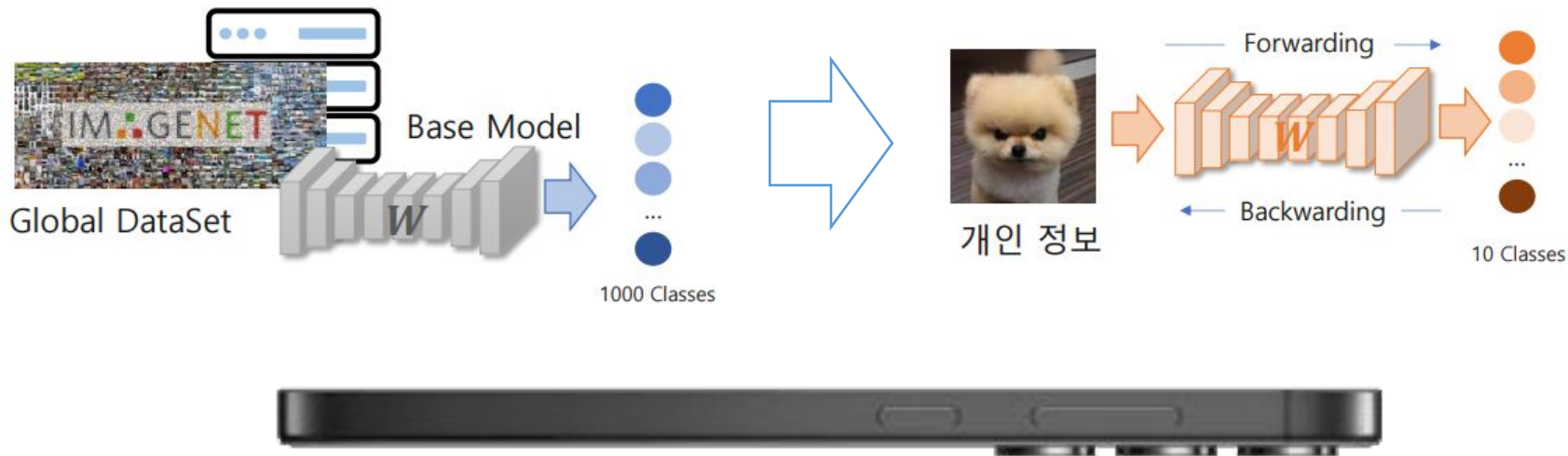


Overview

“Neural Network Trainer”

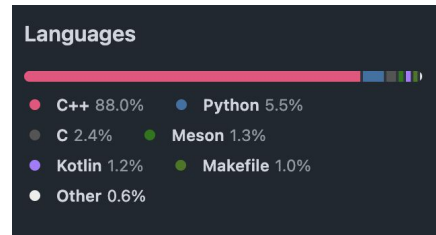


NNTrainer: Lightweight on-device AI framework



What is NNTrainer?

- “A Software Framework for training Neural Network models on devices.”
 - PyTorch/TensorFlow for on-device training
 - Open source, maintained by Samsung Electronics
 - Mostly C++ code with Doxygen
 - Works nicely with NNStreamer
- Focuses on on-device fine-tuning over pre-training
- NNTrainer supports many layers/optimizers/loss functions/... (이하 통칭 “operations” or “ops”)
 - But not all of them....
 - Many things to do!



**Contribute to NNTrainer
By implementing 3 to 5 ops**

Requirement

Some vital ops are missing:

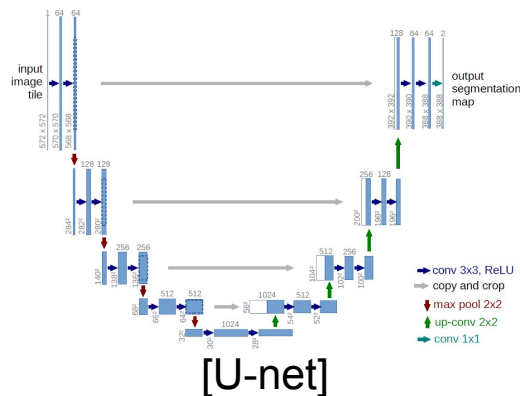
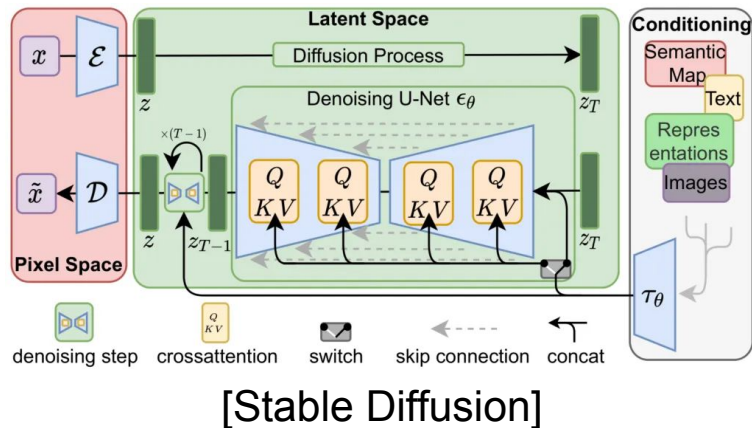
- **ConvTranspose2d**

- Diffusion model 의 기반이 되는 U-net 의 “up-conv” 단계에 사용됨

=> Enables on-device diffusion models
and thus **text-to-image multimodal LLMs**
(Dall-E, Stable Diffusion)



- Ops used in original diffusion model but not implemented in NNTrainer: **LinearLR**, **RMSProp**
- Other unimplemented but popular ops: **CosineAnnealingLR**, **AdamW**



Requirement & Approach

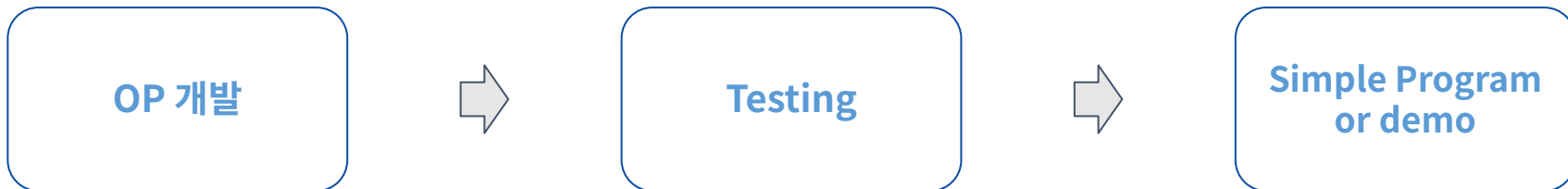
1. U-net 을 목표로!

- a. Implement [ConvTranspose2d](#) (and [LinearLR](#), [RMSProp](#))
- b. With performance and memory efficiency in mind! (On-device 환경을 생각한다.)

2. 그 전에, 연습삼아!

- a. Implement [CosineAnnealingLR](#), [AdamW](#)

개발 사이클



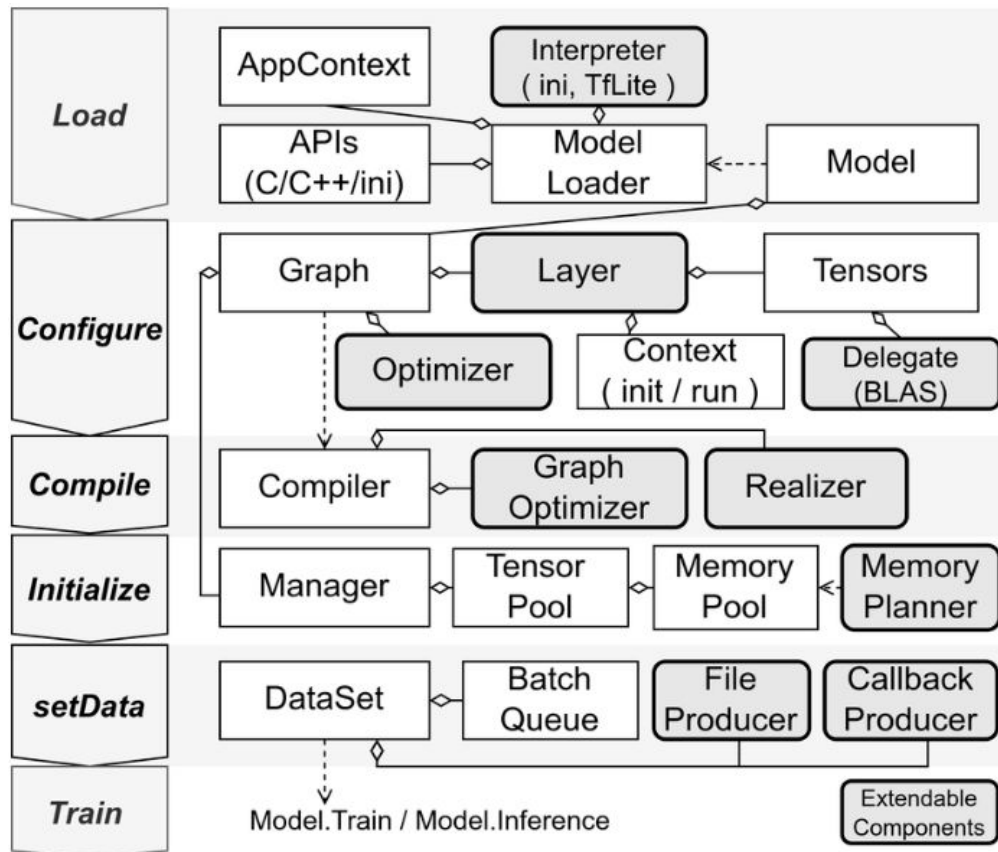
Development Environment

- Any commercial machine running Linux should suffice
 - NNTrainer consists of C++ code, mostly for the CPU
 - Install NNTrainer: [nntainer/docs/getting-started.md](https://nntainer.github.io/docs/getting-started.md) at main
- Collaboration through [Git + GitHub](#)

Dependency list

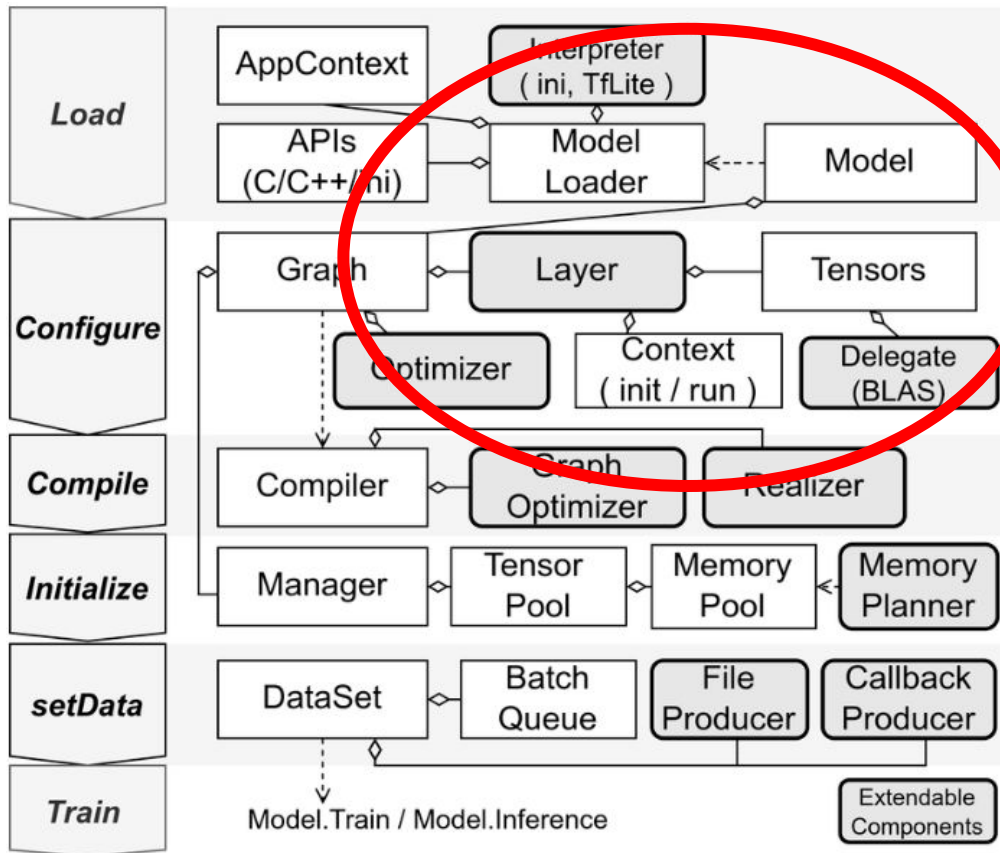
: gcc/g++, meson, libopenblas-dev, tensorflow-lite, libinparser,
libjsonparser, libjsoncpp, libcurl3, libgtest

Architecture



Abstract architecture of NNTrainer

Architecture



Layer들을 개발하면서 주로 contribute 하게 될 부분

- Model
- Model Loader
- Layer
- Optimizer
- ...

Architecture : NNTrainer C++ API를 중심으로

```
int main(int argc, char *argv[]) {
    auto model = create_model();

    model->setProperty({"batch_size=" + std::to_string(batch_size),
                      "epochs=" + std::to_string(epochs),
                      "save_path=" + save_path});

    auto optimizer = ml::train::createOptimizer("SGD", {"learning_rate=" + std::to_string(learning_rate)});
    model->setOptimizer(std::move(optimizer));

    int status = model->compile();
    status = model->initialize();

    auto random_generator = getRandomDataGenerator();
    auto train_dataset = ml::train::createDataset(
        ml::train::DatasetType::GENERATOR, dataset_cb, random_generator.get());

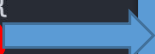
    model->setDataset(ml::train::DatasetModeType::MODE_TRAIN, std::move(train_dataset));

    model->train();

    return status;
}
```

Architecture : NNTrainer C++ API를 중심으로

```
int main(int argc, char *argv[]) {  
    auto model = create_model();  
  
    model->setProperty({"batch_size=" + std::to_string(BATCH_SIZE),  
                      "epochs=" + std::to_string(EPOCHS),  
                      "save_path=" + save_path});  
  
    auto optimizer = ml::train::createOptimizer("adam");  
    model->setOptimizer(std::move(optimizer));  
  
    int status = model->compile();  
    status = model->initialize();  
  
    auto random_generator = getRandomDataGenerator();  
    auto train_dataset = ml::train::createDataset(  
        ml::train::DatasetType::GENERATOR, dataset_cb, random_generator.get());  
  
    model->setDataset(ml::train::DatasetModeType::MODE_TRAIN, std::move(train_dataset));  
  
    model->train();  
  
    return status;  
}
```



```
std::unique_ptr<ml::train::Model> create_model() {  
    std::unique_ptr<ml::train::Model> model =  
        ml::train::createModel(ml::train::ModelType::NEURAL_NET, {"loss=mse"});  
  
    model->addLayer(  
        ml::train::createLayer(  
            "input", {"input_shape=1:1:10"}  
        );  
  
    model->addLayer(  
        ml::train::createLayer("fully_connected", {"unit=5", "activation=softmax"}));  
  
    return model;  
}
```

실제 Layer example (Conv2dLayer)

```
class Conv2DLayer : public LayerImpl {  
    void forwarding(RunLayerContext &context, bool training) override;  
    void calcDerivative(RunLayerContext &context) override;  
    void calcGradient(RunLayerContext &context) override;  
    bool supportBackwarding() const override { return true; }  
}
```

Layer에 대한 Forward, Backward 등을 구현해야 함!

Op 들을 구현하면서 지켜야 할 contribution guide

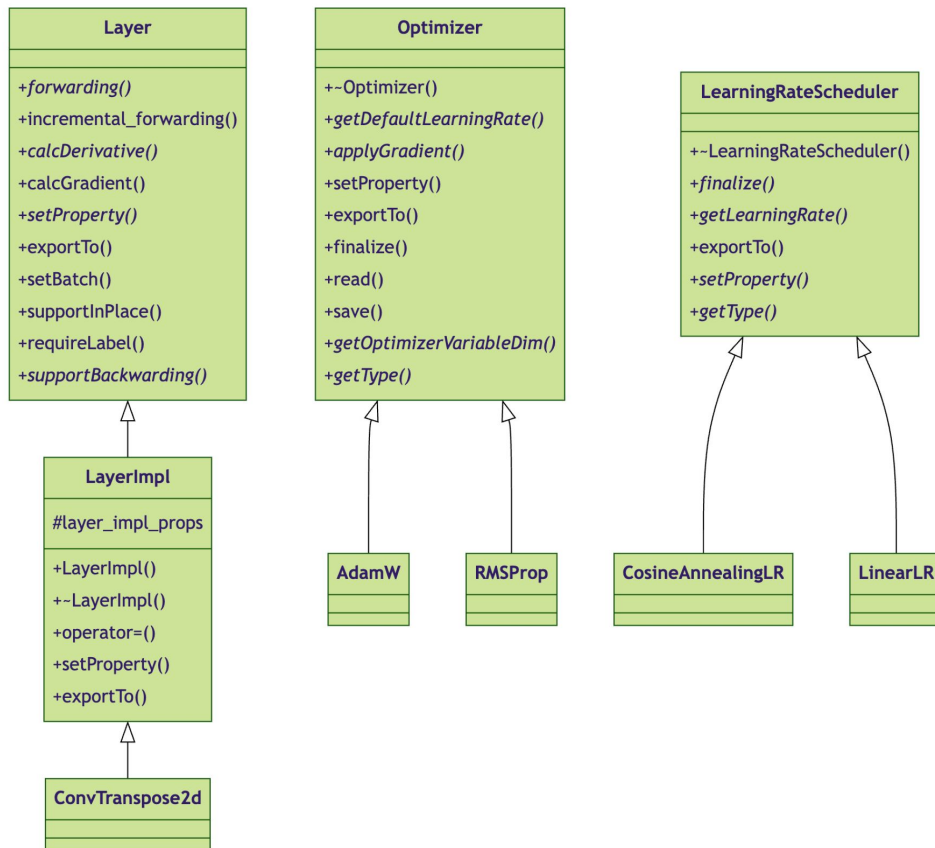
1. Optimizer, scheduler, layer 등 구현
2. 각 feature에 대한 unit test case (GoogleTest or SSAT) 구현
3. 구현한 feature 를 활용하는 simple example (optional) 구현
4. Doxygen을 통한 문서 작성

* Small commits and multiple PRs (회사 담당자님께서 부탁하신 점)

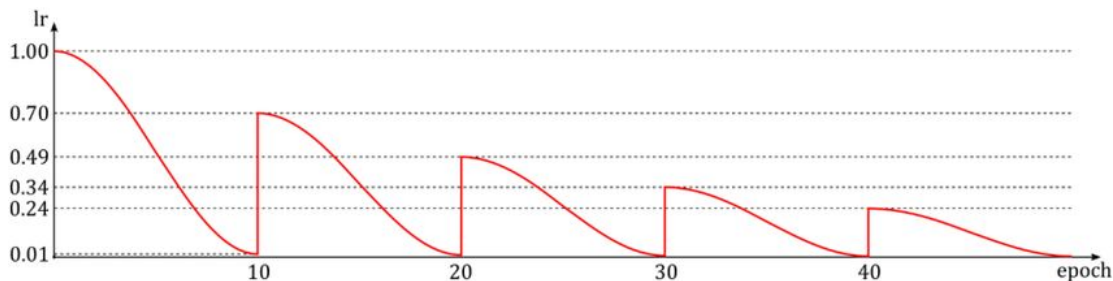
이번 프로젝트 동안 구현할 Op

1. **CosineAnnealingLR** scheduler
2. **AdamW** Optimizer
3. **ConvTranspose2d** Layer
4. **LinearLR** scheduler
5. **RMSProp** optimizer

Implementation Spec : Class Diagram



CosineAnnealingLR Scheduler



Cosine 함수 기반으로 학습률을 조정하는 **러닝 레이트 스케줄러**

해당 스케줄러를 `nntainer`의 `nntainer/optimizers` 에 구현!

AdamW Optimizer

```
for  $t = 1$  to ... do
  if maximize :
     $g_t \leftarrow -\nabla_{\theta} f_t(\theta_{t-1})$ 
  else
     $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ 
   $\theta_t \leftarrow \theta_{t-1} - \gamma \lambda \theta_{t-1}$ 
   $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
   $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
   $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ 
   $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ 
  if amsgrad
     $\widehat{v}_t^{max} \leftarrow \max(\widehat{v}_t^{max}, \widehat{v}_t)$ 
     $\theta_t \leftarrow \theta_t - \gamma \widehat{m}_t / (\sqrt{\widehat{v}_t^{max}} + \epsilon)$ 
  else
     $\theta_t \leftarrow \theta_t - \gamma \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ 
```

Weight decay를 가중치에 직접 적용하는 adam 변형 버전 AdamW optimizer

해당 optimizer를 `nntrainer`의 `nntrainer/optimizers` 에 구현!

ConvTranspose2d Layer

The diagram illustrates the implementation of a ConvTranspose2d layer through four matrix multiplication examples. Each example shows a 2x2 input kernel, a 3x3 weight matrix, and the resulting 4x4 output feature map.

Example 1:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 2 & 1 & 0 \\ 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Example 2:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2+2 & 3+4 & 6 \\ 2 & 2+4 & 1+4 & 2 \\ 3 & 2+6 & 1+4 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Example 3:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 & 6 \\ 2+3 & 6+6 & 5+9 & 2 \\ 3+6 & 8+6 & 5+2 & 2 \\ 9 & 6 & 3 & 0 \end{bmatrix}$$

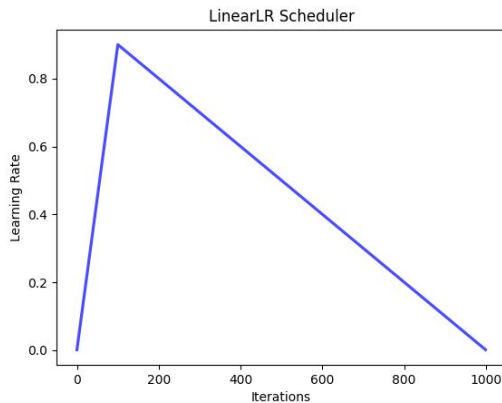
Example 4:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 & 6 \\ 5 & 12+4 & 14+8 & 2+12 \\ 9 & 14+8 & 7+8 & 2+4 \\ 9 & 6+12 & 3+8 & 4 \end{bmatrix}$$

입력 이미지를 업샘플링하는 레이어인 ConvTranspose2d layer

해당 layer를 `nntrainer`의 `nntrainer/layers` 에 구현!

LinearLR Scheduler



학습률을 일정한 비율로 선형적으로 감소시키는 스케줄러

해당 scheduler를 `nntrainer`의 `nntrainer/optimizers` 에 구현!

RMSProp Optimizer

For each Parameter w^j

(j subscript dropped for clarity)

$$\nu_t = \rho \nu_{t-1} + (1 - \rho) * g_t^2$$

$$\Delta \omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta \omega_t$$

η : Initial Learning rate

ν_t : Exponential Average of squares of gradients

g_t : Gradient at time t along ω^j

Gradient에 따라 학습률을 조정하는 적응형 학습률 기반의 optimizer

해당 optimizer를 `nntrainer`의 `nntrainer/optimizers` 에 구현!

Detailed Plan (구체적인 계획)

1. Warm-up (2 ops): CosineAnnealingLR scheduler, AdamW optimizer

- Get to know the internal API
- Performance programming 에 익숙해지기

2. 본격 (1 op): ConvTranspose2d layer

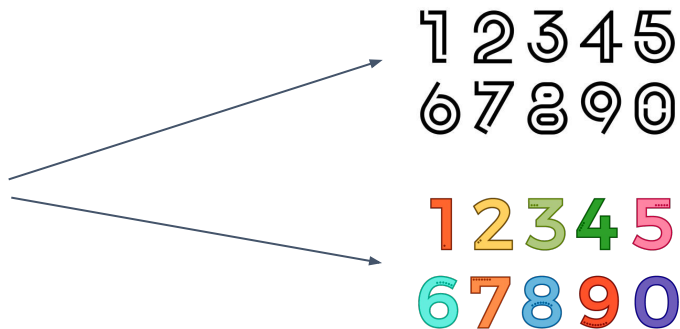
- Implement highly-optimized code leveraging parallelism
- 참고할 수 있는 예시: `fc` and `conv` layers

3. Further (2 ops): LinearLR, RMSProp

- Other ops which the original diffusion model used (for training)

Demo Plan

- Open-source library contribution 의 특성 상 결과를 (시각적으로) 보이기 어려움
- ConvTranspose2d 를 사용하면 NNTrainer 로 U-net, 곧 diffusion model 을 만들 수 있다.
- Op 들을 구현한 후, 가능한 한 MNIST dataset 을 기반으로 한 diffusion model 을 구현하고 학습시켜 demo 할 예정



Division and Assignment of Work

[구현할 것]

CosineAnnealingLR scheduler

AdamW Optimizer

ConvTranspose2D Layer

LinearLR scheduler

RMSProp optimizer

Diffusion model

[구현 중 해야 할 일]

OP 로직 구현

Unit test 구현

최적화 수행

디버깅 및 리팩토링

모델 학습

모델 테스트

[담당]

	CosAnneal LR	AdamW	ConvTrans	LinearLR	RMSProp	Diffusion model
로직 구현	이현우, 장민혁	장민혁, 송우경	송우경, 장민혁	이현우, 장민혁	장민혁, 송우경	장민혁, 이현우
Unit test 구현	이현우	이현우, 장민혁	이현우, 송우경	이현우	이현우, 장민혁	x
최적화 수행	이현우, 송우경	장민혁, 송우경	송우경	이현우, 송우경	장민혁, 송우경	x
디버깅 및 리팩토링	이현우	장민혁	송우경	이현우	장민혁	x
모델 학습	x	x	x	x	x	장민혁, 송우경
모델 테스트	x	x	x	x	x	장민혁, 이현우

Schedule

9월 4주 ~

NNTrainer 스터디

10월 1주 ~

ConsineAnnealingLR, AdamW 구현

10월 4주 ~

ConvTranspose2d 구현

11월 2주 ~

LinearLR, RMSProp 구현

11월 4주 ~

데모 구현 및 wrap-up

감사합니다

발표자

장민혁 | minhyukjang@snu.ac.kr

보완한 점

1. 어떻게 NNTrainer의 OP들일 구현할 것인지 구체적인 다이어그램을 제시(page 17)
 - Layer, Scheduler, Optimizer들을 구현하기 위해 구현해야 하는 function list를 제시
2. 역할 분담 구체화(page 25)
 - 기존 역할 분담을 더욱 구체화해서, 누가 어떤 OP의 어느 부분을 구현할 것인지 구체적으로 작성