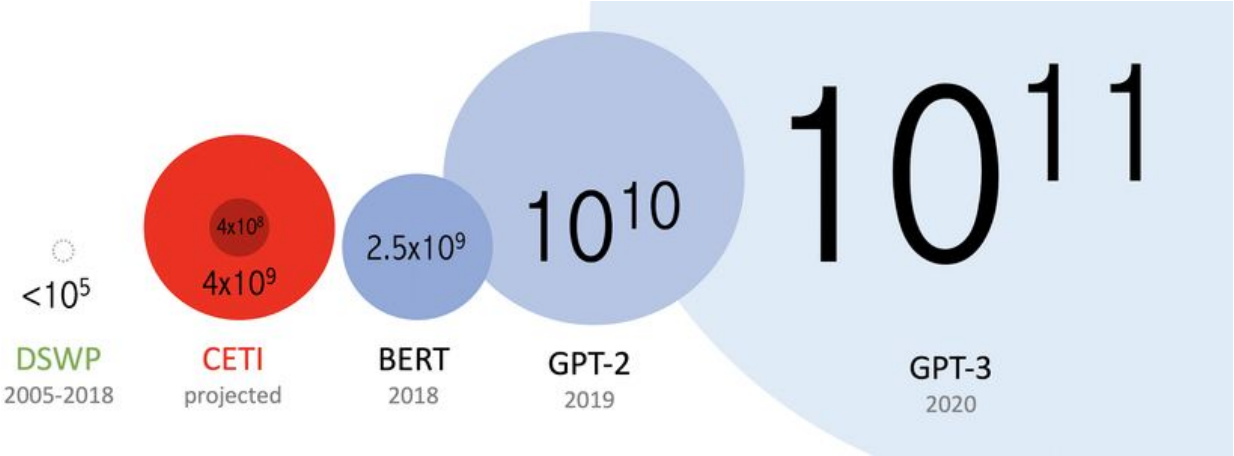
Multi-GPU 학습

TEAMLAB director

최성철

오늘날의 딥러닝은 엄청난 데이터와의 싸움



https://www.researchgate.net/figure/Comparative-size-of-datasets-used-for-training-NLP-models-represented-by-the-circle_fig3_350992250

연구는 장비빨...

http://www.soccer-city.com/news/2015/3/18/urban-and-polished-leo-messi-releases-his-signature-cleat-with-adidas

https://www.nvidia.com/ko-kr/data-center/dgx-systems/

Multi-GPU
어떻게 GPU 다룰 것인가

개념정리

- Single vs. Multi
- GPU vs. Node
- Single Node Single GPU
- Single Node Multi GPU
- Multi Node Multi GPU

https://www.channelpronetwork.com/article/building-ai-and-machine-learning-workstations

https://www.camfil.com/en/industries/electronics-and-optics/data-centers

ANNOUNCING TensorRT 8.0

World-Leading Performance & Accuracy on NVIDIA Ampere Architecture GPUs

2x
Performance vs TensorRT 7

Transformer Optimizations

Sparsity Support on Ampere GPUs

2x
Accuracy vs TensorRT 7 w/ INT8

Quantization Aware Training (QAT)

Available for free to NVIDIA Developer Program members:
developer.nvidia.com/tensorrt

*QAT – Quantization Aware Training, PTQ – Post Training Quantization
TensorRT 8.0: BERT-Large, A100, Precision = INT8 BS=1, Seq Len = 128, A100 | Sparsity: TenorRT 7.2 vs TensorRT 8.0 with Sparsity, BS = 64, Precision = INT8 | QAT: EfficientNet B0

NVIDIA

https://www.phoronix.com/news/NVIDIA-TensorRT-8

boostcamp ai tech

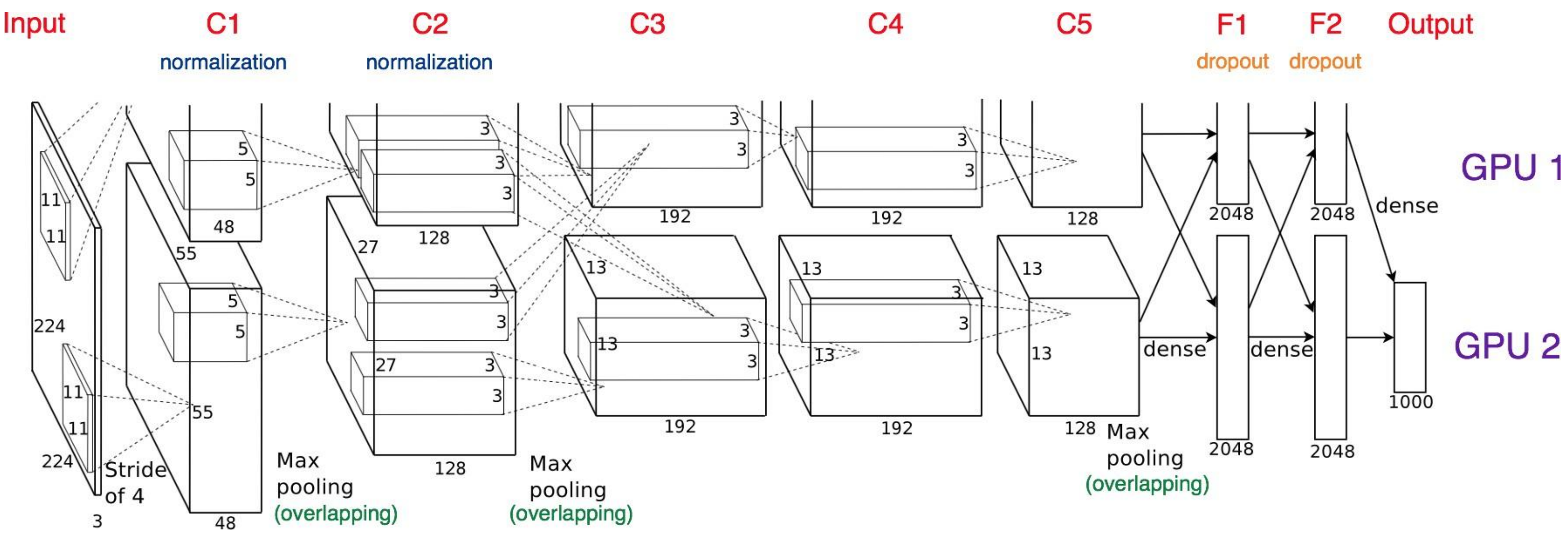
Model parallel

- 다중 **GPU**에 학습을 분산하는 두가지 방법

  모델을 나누기 / 데이터를 나누기
- 모델을 나누는 것은 생각보다 예전부터 썼음 **(alexnet)**
- 모델의 병목, 파이프라인의 어려움 등으로 인해

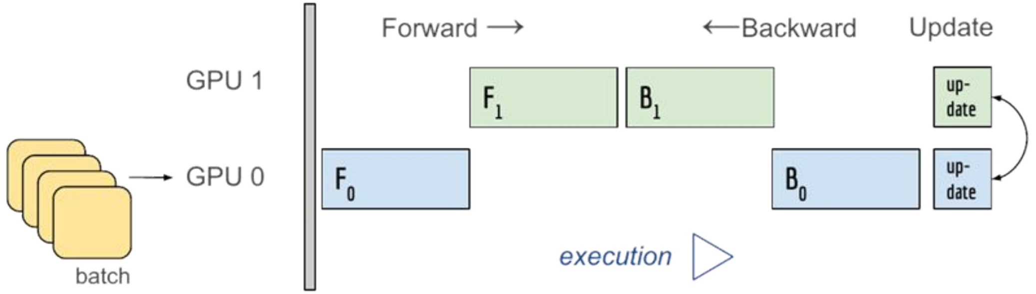
  모델 병렬화는 고난이도 과제

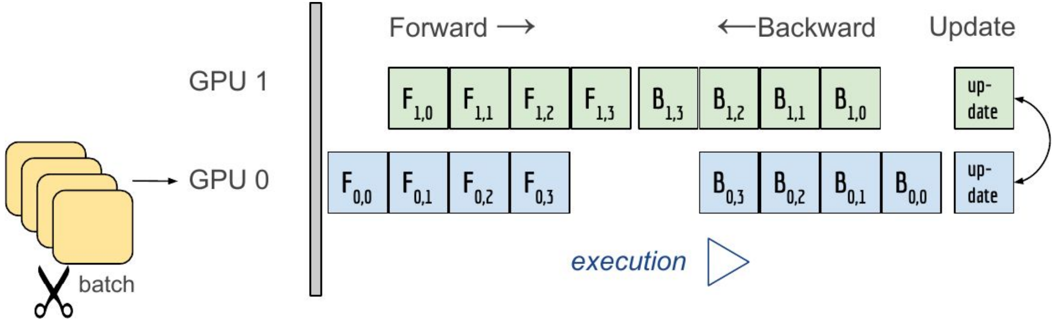
Model parallel vs Data parallel



https://people.cs.pitt.edu/~mzhang/cs1699/hw4.html

Model parallel





http://www.idris.fr/eng/ia/model-parallelism-pytorch-eng.html

Model parallel


```
class ModelParallelResNet50(ResNet):
    def __init__(self, *args, **kwargs):
        super(ModelParallelResNet50,self).__init_(
            Bottleneck, [3, 4, 6, 3], num_classes=num_classes, *args, **kwargs)

        self.seq1 = nn.Sequential(
            self.conv1, self.bn1, self.relu, self.maxpool, self.layer1, self.layer2
        ).to('cuda:0')

        self.seq2 = nn.Sequential(
            self.layer3, self.layer4, self.avgpool,
        ).to('cuda:1')

        self.fc.to('cuda:1')

    def forward(self, x):
        x = self.seq2(self.seq1(x).to('cuda:1'))
        return self.fc(x.view(x.size(0), -1))
```

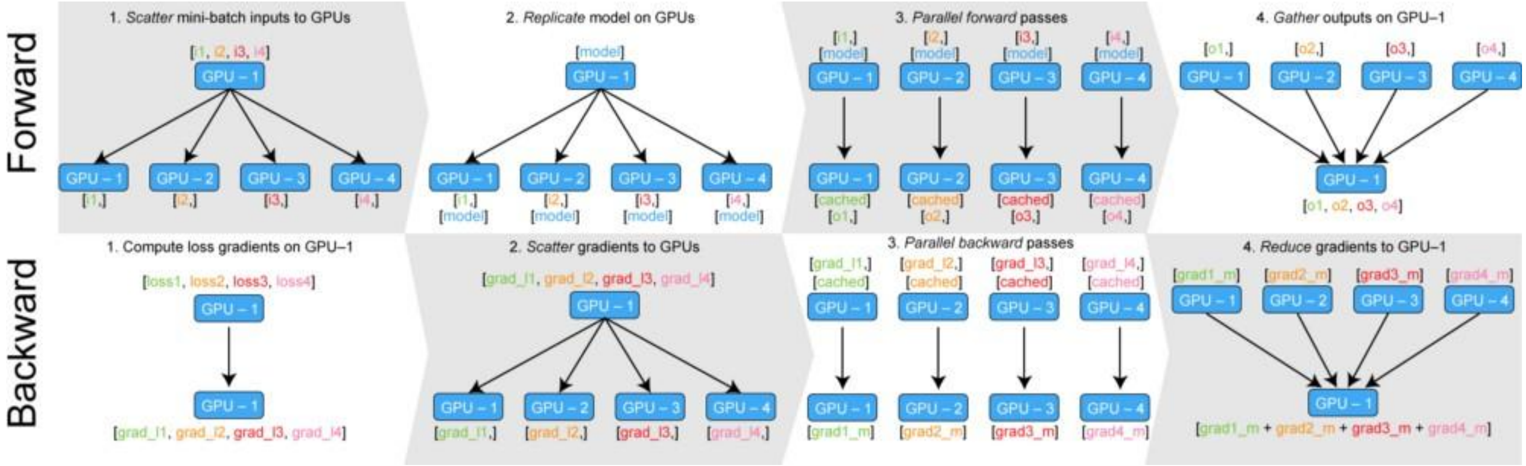
첫번째 모델을 cuda 0에 할당

두번째 모델을 cuda 1에 할당

두 모델을 연결하기

Data parallel

- 데이터를 나눠 **GPU**에 할당후 결과의 평균을 취하는 방법
- **minibatch** 수식과 유사한데 한번에 여러 **GPU**에서 수행



https://bit.ly/37usURV

Data parallel

- **PyTorch에서는 아래 두 가지 방식을 제공**

  DataParallel, DistributedDataParallel

- DataParallel - **단순히 데이터를 분배한 후 평균을 취함**

  → **GPU 사용 불균형 문제 발생, Batch 사이즈 감소 (한 GPU가 병목), GIL**

- DistributedDataParallel - **각 CPU마다 process 생성하여 개별 GPU에 할당**

  → **기본적으로 DataParallel로 하나 개별적으로 연산의 평균을 냄**

Data parallel


```
parallel_model = torch.nn.DataParallel(model) # Encapsulate the model
```

이게 전부...

```
predictions = parallel_model(inputs).      # Forward pass on multi-GPUs
loss = loss_function(predictions, labels)  # Compute loss function
loss.mean().backward()          # Average GPU-losses +backward pass
optimizer.step()         # Optimizer step
predictions = parallel_model(inputs)  # Forward pass with new parameters
```

https://bit.ly/37usURV

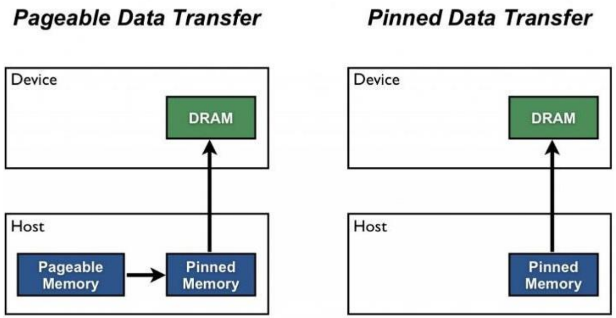
DistributedDataParallel

```
train_sampler = torch.utils.data.distributed.DistributedSampler(train_data)
shuffle = False
pin_memory = True

trainloader = torch.utils.data.DataLoader(train_data, batch_size=20, shuffle=True
                                          pin_memory=pin_memory, num_workers=3,
shuffle=shuffle, sampler=train_sampler)
```

Sampler를 사용

pin_memory = True



DistributedDataParallel


```python
def main():
        n_gpus = torch.cuda.device_count()
        torch.multiprocessing.spawn(main_worker, nprocs=n_gpus, args=(n_gpus, ))

def main_worker(gpu, n_gpus):
    image_size = 224
    batch_size = 512
    num_worker = 8
    epochs = ...

    batch_size = int(batch_size / n_gpus)
    num_worker = int(num_worker / n_gpus)

    torch.distributed.init_process_group(
                backend='nccl', init_method='tcp://127.0.0.1:2568', world_size=n_gpus, rank=gpu)

    model = MODEL

    torch.cuda.set_device(gpu)
    model = model.cuda(gpu)
    model = torch.nn.parallel.DistributedDataParallel(model, device_ids=[gpu])
```

멀티프로세싱 통신 규약 정의

Distributed DataParallel 정의

```python
from multiprocessing import Pool

def f(x):
        return x*x

if __name__ == '__main__':
        with Pool(5) as p:
          print(p.map(f, [1, 2, 3]))
```

Python의 멀티프로세싱 코드

End of Document
Thank You.