**고급소프트웨어실습**

**기말 리포트**

**(CSE 4152)**

**Due: 2020년, 12/17일, 오후 12시 (정오)**

**학번 :20150325**

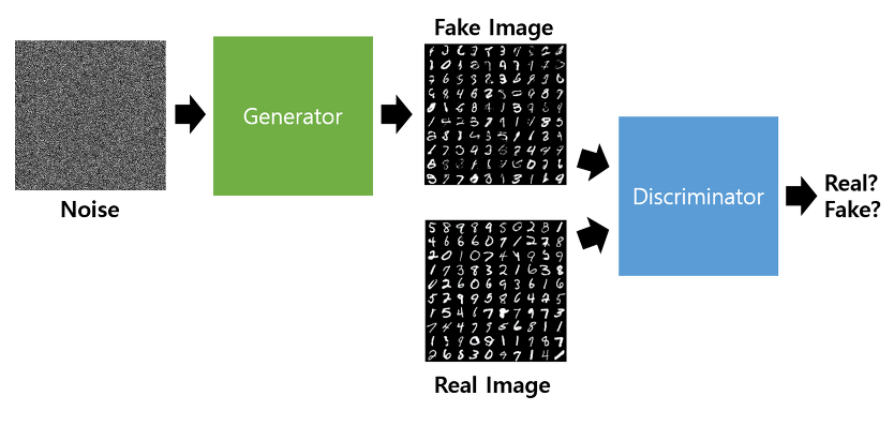
**반 번호 :4**

**이름 :김정수**

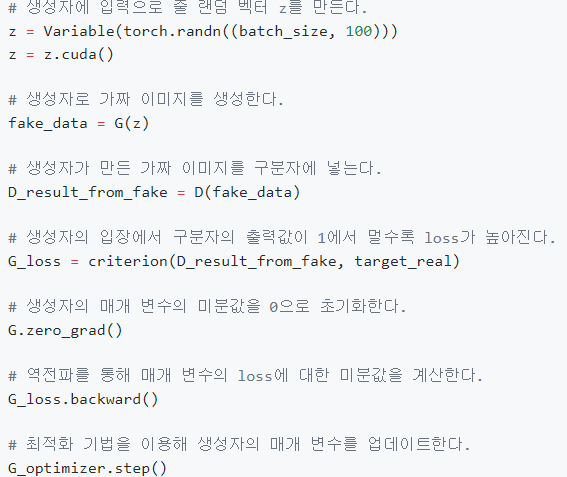
1. **영상 처리 분야에서 Deep Neural Network (DNN) 기술을 사용하여 상용화된 SW 또는 제품의 사례가 있는지 조사하고, 있다면 대표적인 한 가지에 사용된 DNN 기술에 대하여 구조도, 기능, 특징을 포함하여 한 페이지 이내로 설명하시오. 만약 상용화 단계의 사례가 없다면, 연구 개발 등에 많이 활용되는 대표적인 영상 처리 DNN 기술에 대하여 같은 방식으로 설명하시오.**

**영상 처리 DNN기술은 현재 많은 분야에서 연구 개발에 사용되고 있다. 그 중 가장 적극적으로 이에 대해 기술하고 있는 곳은, 의료분야라고 할 수 있다. 대표적으로, 한 예시를 꼽자면, 한 연구에서 약 120여 개의 층을 기반으로 한 cnn은 전두 흉부 x-ray 이미지에서 폐렴을 검출하는 작업에서 전문의보다 더 우수한 정확도를 보였다고 한다.**

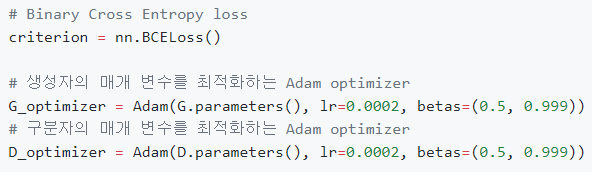
**이러한 DNN기술 중, 수업시간에 중점적으로 다룬 CNN을 제하고 연구 개발에 활용되는 또다른 영상처리 DNN기술은 GAN(Generative Adversarial Network, 생성적 적대신경망)가 있다. 이는 인간의 개입이나 학습 데이터가 없어도, 스스로 학습하는 서로 대립적인 시스템이, 하나가 생성자(G), 하나는 판별자(D)가 되어 경쟁하는 방식으로 학습을 한다. 이런 적대적 학습과정 중 생성자는 결과물을 내면, 판별자에 의해 실제 데이터와 구분된다. 가령, 이미지를 생성하는 AI와 이를 판별하는 AI가 경쟁하면서, 가짜 이미지를 진짜 이미지에 최대한 가깝게 생성해내는 것이다.**

****

**이제 구조를 생각해보자. D는 진짜 이미지에선 1에 가까운 값을, 가짜 이미지에선 0에 가까운 값을 출력해야 한다. 따라서 D의 손실함수는 진짜의 출력값과 1의 차, 가짜의 출력값과 0의 차, 두 경우의 합을 손실함수로 갖는다. 따라서 이를 최소화시켜야 한다. 반면, 생성자G는 랜덤벡터 z(위의 그림의 Noise)를 입력으로 받아, 가짜 이미지를 출력하고, 이를 D에 넣었을 때 1이 출력되도록 해야 한다. 이 값이 1에서 떨어진 정도가 생성자의 손실 함수가 될 것이며, 이를 최소화시켜야 한다. 따라서, G와 D의 대략적인 구조는 아래와 같다.**

****

**그리고 이 D와 G의 매개변수를 업데이트하는 최적화함수는, Adam이 주로 쓰인다.**

**이때 두 번째 인자인 lr은 learning rate를 뜻한다. 위 함수들을 반복하면서, D와 G는 서로 경쟁하면서 학습을 하게 된다.**

**이러한 GAN은 현재 많은 기대를 받고 있는 모델이지만, 아직 여러 한계점이 존재하는데, 그 중 대표적인 것은 바로 학습이 어렵다는 것이다. 즉, 양 유닛이 서로 균형을 맞춰 발전해야 하는데, 만일 하나가 급격히 발전하면 다른 하나는 학습을 진행할 수 없게 되고, 전체적인 학습이 멈춰버린다. 이를 모드 붕괴(Mode Collapse)라 한다.**

**그럼에도 불구하고, 현재 GAN은 여러 분야에서 다른 방식으로 개조되어 응용되고 있다 대표적으로, EnhanceNet은 GAN의 손실함수를 적용해, 저해상도의 이미지나 영상을 고해상도로 변환하는 Super Resolution의 성능을 높였다. 즉, G의 인풋으로 noise가 아닌 저해상도의 이미지를 넣어, D를 진짜 고해상도로 속일 때까지 해당 이미지를 고해상도로 학습시키는 것이다. 이외에도 가상의 얼굴 영상을 생성하는 styleGAN과 같이, GAN은 현재 수많은 영상처리 분야에서 획기적인 DNN기술로 주목받고 있다.**

**2. 아래의 (CODE 1)은 다음과 같은 C/C++ 코드를 어떤 이전 버전의 Visual Studio의 DEBUG 모드에서 컴파일하여 생성한 어셈블러 코드의 일부이다.**

switch (val) {

case 0: tmp = val\*misc; break;

case 1: tmp = val + misc; break;

case 2: tmp = val/misc; break;

**/\* here \*/**

}

**위의 C/C++ 코드의 /\* here \*/ 부분에 다음과 같은 문장을 추가한 후 동일 조건에서 컴파일하였을 때, (CODE 2)와 같은 결과를 얻었다.**

case 3: tmp = val - misc; break;

**수행 속도 차원에서 어떠한 이유로 (CODE 2)의 계산 방식이 (CODE 1)의 방식보다 우월할 수 있는지, 각 방법의 작동/코드 수행 원리를 반드시 설명하면서 상세히 기술하라.**

|  |  |
| --- | --- |
| 그림입니다.  원본 그림의 이름: CLP000008f4731e.bmp  원본 그림의 크기: 가로 473pixel, 세로 733pixel | 그림입니다.  원본 그림의 이름: CLP000008f40001.bmp  원본 그림의 크기: 가로 485pixel, 세로 858pixel |

**위의 어셈블리 코드에서, CODE1의 L544는 CODE2의 L546에 대응된다. Imul, 즉 곱셈을 하는 곳으로, 사용하는 레지스터만 다를 뿐, 나머지는 전부 같다는 것을 알 수 있다. 마찬가지로 CODE1의 L545는 CODE2의 L547에 대응되고, 더하기(add)를 하는 곳이며, CODE1의 L546는 CODE2의 L548과 대응된다. 여기서는 idiv, 즉 나누기를 실행한다. 오직 다음 L549를 진행하지 않고 L543으로 넘어가기 위한 JMP만 추가되었을 뿐이다. 따라서 CODE1과 CODE2의 차이점은 오직 L549과 L568의 존재유무와, 최상단의 코드이다. 이중, L549는 CASE3이 들어올 때를 대비한 단순한 뺄셈 함수(SUB을 이용한)이므로, 진정 CODE1과 2의 차이점은 L568과 최상단 코드에서 나타난다고 할 수 있다.**

**이때 L568의 DD는 DOUBLE WORD, 즉 32비트(=4바이트)를 의미한다. 각 L, 라벨은 DOUBLE WORD로 DEFINE된 것이다. 이제, 이 L568의 DOUBLE WORD 라벨링과 최상단 코드가, 어떻게 CODE1에 비해 빠른 속도를 낼 수 있는 것인지 분석해보겠다.**

**CODE1의 최상단을 보자. 첫 번째 줄에서, EAX레지스터에 VAL값을 받은 다음, 이를 다음 줄에서 다시 DWORD PTR – 8+[EBP]에 넣는다. 이후로 총 3번의 CMP, JE가 나오고, 마지막에 JMP L541이 나온다. 즉, 해당 값을 각각 3번씩, 0부터 2까지를 비교한 다음에 알맞은 라벨로 넘기는 것이다. 여기에서, 만약 VAL이 0~2가 아니라면, 의미없는 CMP를 총 3번 거치게 되며, 2라 할지라도 CMP를 의미없이 2번이나 거치게 된다.**

**반대로 CODE2의 최상단을 보자. 여기에선 CMP DWORD PTR – 8+[EBP], 3과 JA를 함께 쓴다. 따라서, 만약 3보다 큰 값이 나올 경우엔 바로 L543으로 JMP함으로써, 위처럼 적절한 값이 나올 경우 여러 번의 CMP를 거치느라 필요한 오버헤드를 획기적으로 감소시켰다. 다음으로, ECX에 해당 값을 넣은 다음, JMP DWORD PTR &L568[ECX\*4]를 실행한다. L568에는 각 라벨이 DD형태로 DEFINE되어있는데, ECX는 1BYTE이다. 즉, ECX가 0이라면 0\*4=0이니, 바로 L568의 0번째 OFFSET에 있는, L546을 실행하게 될 것이고, 3이라면 3\*4=12이니, L568의 12번째 OFFSET에 있는 L548을 실행할 것이다.**

**즉, 예상 외 VAL에서는 연산이 CMP는 3번에서 1번으로 줄었고, 그 외 모든 경우에서는 예외처리인 CMP 1번에 더해 MOV와 JMP연산만 실행하면 되니, 비록 VAL==1일 때엔 연산 회수가 기존의 단 한번의 CMP을 하는 것에 비해 증가했지만 다른 케이스에서는 연산이 같거나 더 감소하였으니, 평균적으로는 그 방식이 더 나아졌다고 할 수 있다.**

**3. CUDA memory hierarchy에 대하여 아래한글 기준 1500글자(공백 제외) 이상의 분량으로 명료하게 기술하라. 강의자료의 내용을 copy-and-paste할 경우 답으로 인정하지 않으며, 반드시 자신의 언어로 CUDA에 대하여 잘 모르는 컴퓨터공학 전공 3학년에게 설명하는 수준에서 기술할 것. (제출 내용에 대하여 copy-check를 진행할 예정임)**

**CUDA(COMPUTE UNIFIED DEVICE ARCHITECTUR)란, 그래픽 처리장치(GPU)에서 수행하는 병렬처리 알고리즘을, C프로그래밍 언어와 같은 표준 언어 등을 이용해 작성할 수 있도록 하는 GPGPU(gpu를 비그래픽 애플리케이션에 응용하려는 시도)기술이다. GPU는 ALU(ARITHMETIC LOGIC UNIT, 산술논리장치)가 CPU보다 많기에, CPU보다 더 빠른 연산이 하지만 CPU와 달리 ALU를 일일이 지정해 실행해야 한다. 따라서, 이를 위해 CUDA프로그래밍을 위해선 이 구조, 즉 cuda 메모리 계층구조(cuda memory hierarchy)에 대해 알아야 할 필요가 있다.**

**CUDA는 스레드->블록->그리드로 구조가 계층적으로 되어 있으며, 각각 레지스터-공유메모리-전역메모리(그 외 상수, 텍스쳐 메모리)를 제공한다. 따라서 스레드는 자신만의 로컬메모리를 가지며, 한 블록 내의 스레드는 서로 공유하는 shared memory가 있고, 블록의 집합인 그리드 역시 데이터 공유를 위한 글로벌메모리가 존재한다. 또한, 스레드로 갈수록 용량이 작은 대신 액세스가 빠르고, 그리드로 갈수록 용량이 커지며 액세스가 느려진다.**

**스레드는 실제 일을 실행하는 최소 단위이다. 상술했듯 cuda에서는 ALU를 지정해야 하므로, 스레드를 쓰기 위해서는, 가령 N번 그리드의 M번 블록의 L번 스레드를 쓰겠다는 지정이 필요하다. 이 스레드는 여러 개가 묶여 1개의 코드 진행 단위, 즉 병렬처리의 수행단위인 워프(warp, 또는 스트리밍 멀티프로세서(sm))로 구성될 수 있다. 구현에 따라 다르지만, 현재는 보통 32개로 묶인다.**

**워프는 병렬적으로 수행되며, 스케줄러는 이 워프를 바꿔가며 실행한다. 또한 이 워프는 여러 개의 스트림 프로세서(sp, 또는 cuda core, 보통 4개의 스레드)를 갖는다. 즉, 하나의 sm당 약 8개의 sp를 갖는다. 워프는 각각 독립적 명령어 스케쥴러를 갖춰, 다른 워프의 cuda코어와 독립적으로, 여러 개의 스레드를 동시에 실행한다. 각 sp는 기본적인 논리연산을 수행한다. 이렇게 워프로 묶는 장점으로는, 프로그래밍 도중 커널 코드가 메모리 접근과 같은, 소모시간이 큰 작업을 하는 동안 다른 워프로 스위칭 해 연산을 수행하는 것이 가능하다. 이를 ‘latency hiding’이라 한다. 또다른 warp의 장점으로는, 동일 워프의 스레드는 전역 메모리 접근 시, 메모리 통합 접근이 가능하여, 메모리i/o 접근 지연시간이 크게 단축된다.**

**이 스레드가 모인 블록은, 종류에 따라 512개나 최대 1024개까지의 스레드를 갖는다. 블록은 일종의 스레드를 관리하기 위한 인덱싱 방식이며, 사용자의 편의에 따라 1차원으로 구성될 수도, 혹은 3차원으로 구성될 수 있다. 각 블록의 스레드는 해당 블록의 idx(blockIdx)와 차원(blockDim)를 곱하고, 여기에 접근하고자 하는 스레드의 인덱스(threadId)를 더함으로써 접근할 수 있다.**

**마지막으로 그리드는 커널이 실행될 때, 스레드가 구성되는 최상위 집합체이다. 그 하나가 디바이스에서 커널이 실행되는 응용프로그램 자체를 나타내며, 따라서 하나의 디바이스(=gpu)에서 실행되는 그리드는 한 시점에서 오직 하나이다. 즉, 한 디바이스는 동시에 2개 이상의 그리드를 돌릴 수 없다.**

**결론적으로, 위의 구조를 이용해 프로그램을 최적화시키면 프로그램의 성능을 끌어올릴 수 있다. Gpgpu프로그램을 수행하는 기본단위는 워프이며, gpu는 워프의 집합이다. 활성화된 스레드가 많을 수록, 메모리 참조 등으로 지연시간이 발생할 때 다른 활성화된 스레드가 다른 연산을 수행함으로써 워프의 쉬는 시간이 감소하고, 성능이 올라간다. 한 블록은 하나의 워프에서 처리되며, 이 워프에는 레지스터나 스레드, shared memory등이 할당되어있다. 따라서, 하나의 스레드가 사용하는 레지스터의 수와 워프 당 레지스터의 수, 사용하는 공유 메모리의 크기를 통해 활성화된 블록의 수를 예측한 다음, 활성화된 블록의 수를 최대한으로 늘리는 것이 사용하는 워프의 수를 늘리고 성능을 높여, 데이터-병렬의 계산양이 많은 함수, 가령 for문 같은 것의 성능을 높이는 방법이 된다.**

**============= 글자수(공백 제외): 1502 =============**

**역사(歷史, 문화어: 력사, 영어: history)는 인류 사회의 변천과 흥망의 과정, 지난 시대에 남긴 기록물, 이를 연구하는 학문 분야 등을 가리킨다. 또 인간이 거쳐온 모습이나 인간의 행위로 일어난 사실을 말하는 단어로도 쓰인다. 또한 역사는 시간의 흐름으로써 어떤 사람이 겪은 일에서 중요한 일들 중 후대에게 쉽게 잊히지 않는 과거의 사실을 의미하며 명예와는 미래에서 잊히지 않고 이어 전해진다는 차이가 존재한다.[1][2] 역사라는 말은 객관적 사실과 서정적 표현, 주관적 기술의 세 측면의 의미를 가지는 것으로 정의되고 있는데, 레오폴트 폰 랑케는 "있었던 그대로의 과거"를 밝혀내는 것이 역사가의 사명이라고 하여 객관적 사실을 강조하였다. 이에 비해 에드워드 핼릿 카는 과거의 사실을 보는 역사가의 관점과 사회 변화에 따라 역사가 달리 쓰일 수 있다고 하였다. 역사 연구란 이야기(narrative)를 통해 연속된 사건들을 검증하고 분석하며 종종 인과 관계를 객관적으로 규정하기도 한다. 역사학은 고고학, 사회학 등의 주변 학문과 밀접한 연관을 갖고 있다. 역사라는 개념은 또한 넓은 의미에서 사건이나 사물의 자취를 총칭하는 데 쓰이기도 한다. 역사학자들은 과거를 연구하기 위하여 많은 역사적 자료를 탐색한다. 이 자료들은 1차 사료와 2차 사료로 구분되는데, 1차 사료는 어떤 사건에 직접 참여한 사람들이 남긴 문서, 기록 등이다.**

**역사(歷史, 문화어: 력사, 영어: history)는 인류 사회의 변천과 흥망의 과정, 지난 시대에 남긴 기록물, 이를 연구하는 학문 분야 등을 가리킨다. 또 인간이 거쳐온 모습이나 인간의 행위로 일어난 사실을 말하는 단어로도 쓰인다. 또한 역사는 시간의 흐름으로써 어떤 사람이 겪은 일에서 중요한 일들 중 후대에게 쉽게 잊히지 않는 과거의 사실을 의미하며 명예와는 미래에서 잊히지 않고 이어 전해진다는 차이가 존재한다.[1][2] 역사라는 말은 객관적 사실과 서정적 표현, 주관적 기술의 세 측면의 의미를 가지는 것으로 정의되고 있는데, 레오폴트 폰 랑케는 "있었던 그대로의 과거"**

**===================================================================**