실전 기계학습
기말 프로젝트

2022105780 손정우

2019102212 이정호

2022105773 김재욱

경희대학교
KYUNG HEE UNIVERSITY

목차

경희대학교
KYUNG HEE UNIVERSITY

학습동기

이미지 복원 학습 동기

: 현재 인공지능의 많은 task에서 이미지 복원을 통해 성능 향상 됨

모델 변천 과정

1. DnCNN(28.5)
2. SRResNet (4.7)
3. #Non - AI
4. Pix2Pix (non)
5. UNet (4.8)
6. KBNet (3.4)

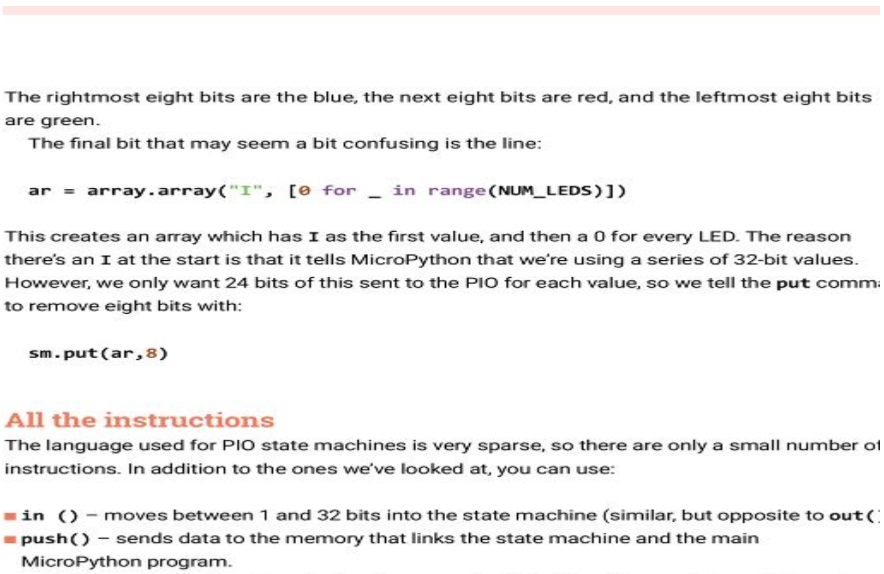
모델 순위

1. KBNet

2. SRResNet

1. SRResNet

기본 모델 선택: SRResNet



The rightmost eight bits are the blue, the next eight bits are red, and the leftmost eight bits are green.

The final bit that may seem a bit confusing is the line:

```
ar = array.array("I", [0 for _ in range(NUM_LEDS)])
```

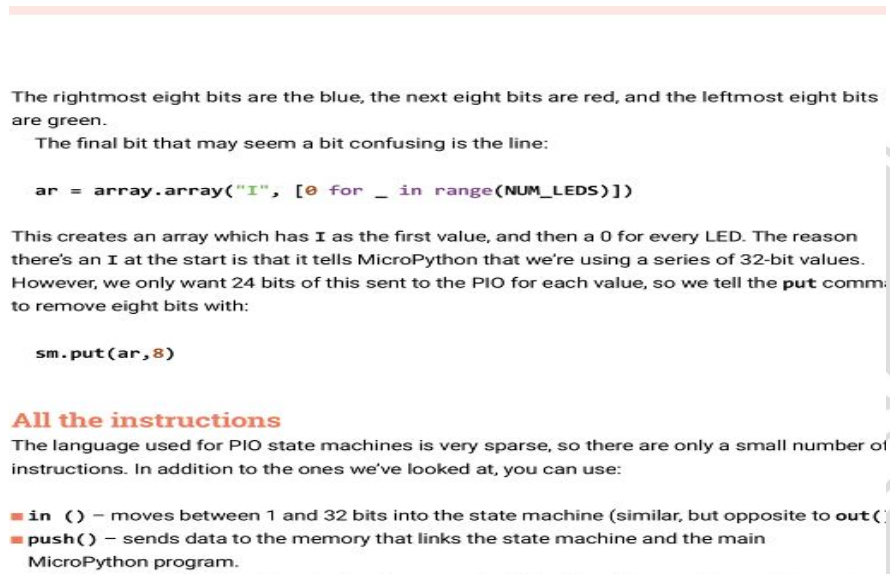
This creates an array which has I as the first value, and then a 0 for every LED. The reason there's an I at the start is that it tells MicroPython that we're using a series of 32-bit values. However, we only want 24 bits of this sent to the PIO for each value, so we tell the **put** comm to remove eight bits with:

```
sm.put(ar,8)
```

All the instructions

The language used for PIO state machines is very sparse, so there are only a small number of instructions. In addition to the ones we've looked at, you can use:

- **in ()** – moves between 1 and 32 bits into the state machine (similar, but opposite to **out(**
- **push()** – sends data to the memory that links the state machine and the main MicroPython program.



The rightmost eight bits are the blue, the next eight bits are red, and the leftmost eight bits are green.

The final bit that may seem a bit confusing is the line:

ar = array.array("I", [0 for _ in range(NUM_LEDS)])

This creates an array which has I as the first value, and then a 0 for every LED. The reason there's an I at the start is that it tells MicroPython that we're using a series of 32-bit values. However, we only want 24 bits of this sent to the PIO for each value, so we tell the put comm to remove eight bits with:

sm.put(ar,8)

All the instructions

The language used for PIO state machines is very sparse, so there are only a small number of instructions. In addition to the ones we've looked at, you can use:

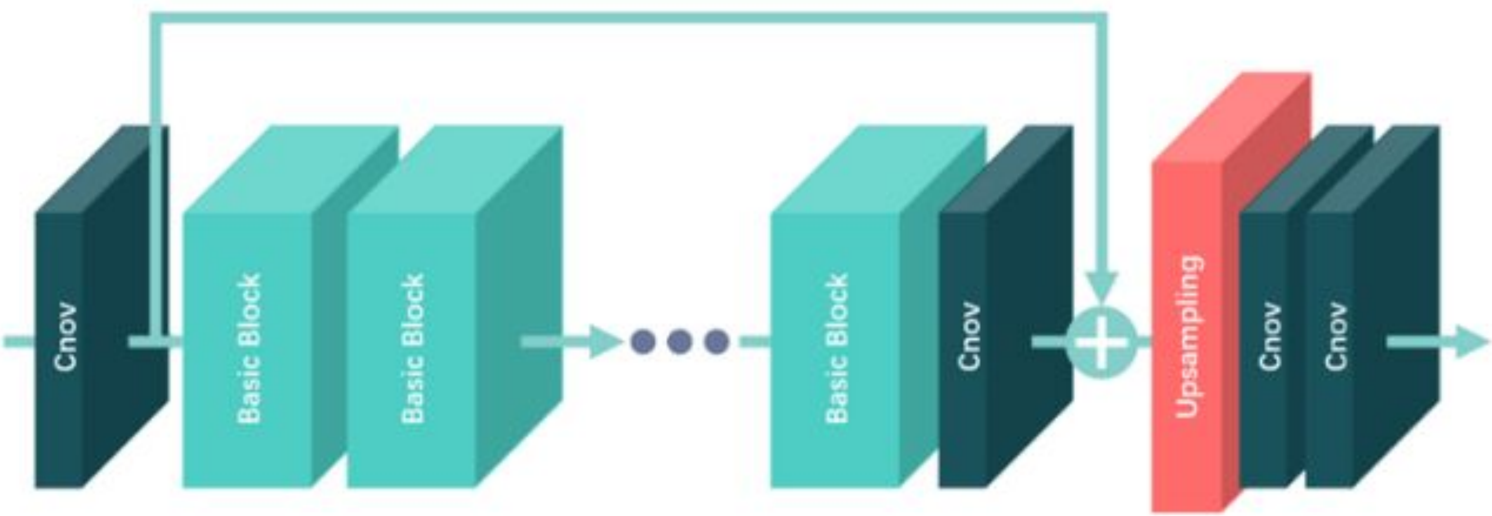
in () – moves between 1 and 32 bits into the state machine (similar, but opposite to out(
push() – sends data to the memory that links the state machine and the main MicroPython program.

- (새로 데이터셋 주어지기 이전) 정확한 데이터의 결함을 알고자 데이터의 노이즈를 분석하고자 확인하였지만, 딱히 육안으로 노이즈 보이지 않음

>> 따라서, 해상도를 높이기 위해 모델 검색

경희대학교
KYUNG HEE UNIVERSITY

SRResNet



이론적 배경 : 잔차블록, 업샘플링 계층

학습 계획

1. 단순한 모델 구조(블럭 구조, 층 개수)
2. 잔류 블록의 구조 따른 성능 변화
3. 정규화의 다양한 방법
4. 다양한 손실함수
5. augmentation

1차 시도

1. 블럭의 구조 변화 : 3X3+3X3 -> 1X1+3X3+1X1, 실패
2. loss함수 변경: mse > mae > perceptual loss, 실패
3. 활성함수 변경: relu -> prelu -> gelu -> GLU, 실패
4. transform부분 변경을 통해 augmentation 시도, 실패

경희대학교 KYUNG HEE UNIVERSITY

1차 시도 결과

학습 결과

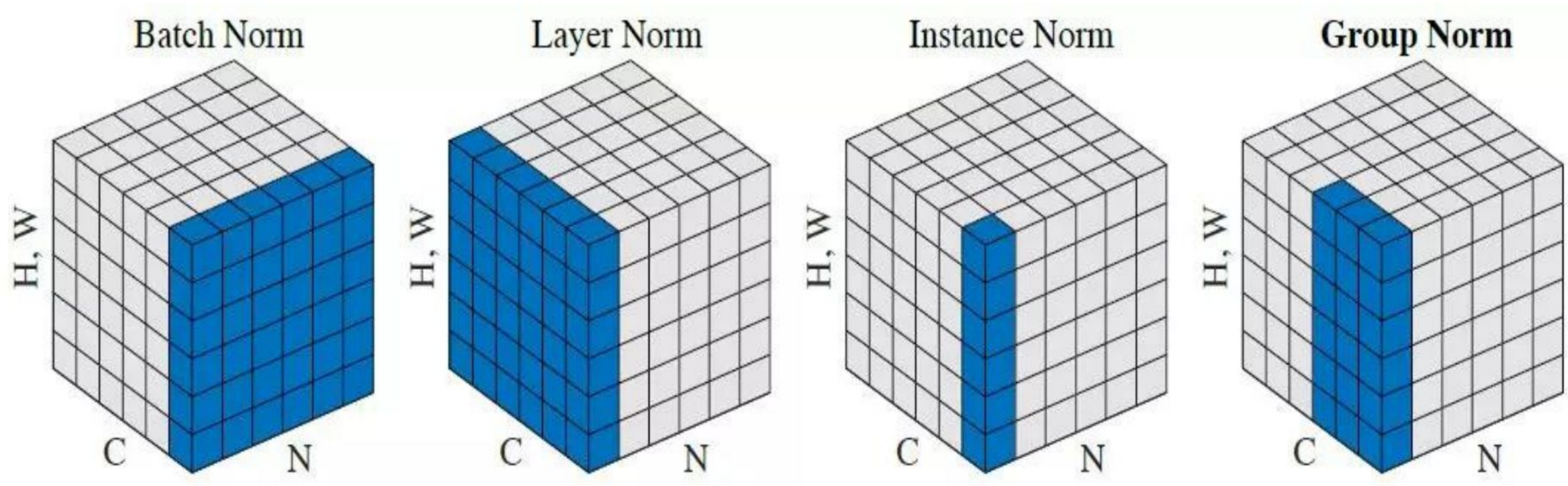
Train Score : 0.03

Val Score : 0.04

kaggle score : 28점

성능 변화 X

2차 시도



- 다양한 정규화 방법 시도

2차 시도

```
train_transform = Compose([
    ToTensor()
])
```

- 채널 정규화 부분 삭제

2차 시도 결과

학습 결과

Train Score : 0.006

Val Score : 0.007

kaggle score : 4.74점

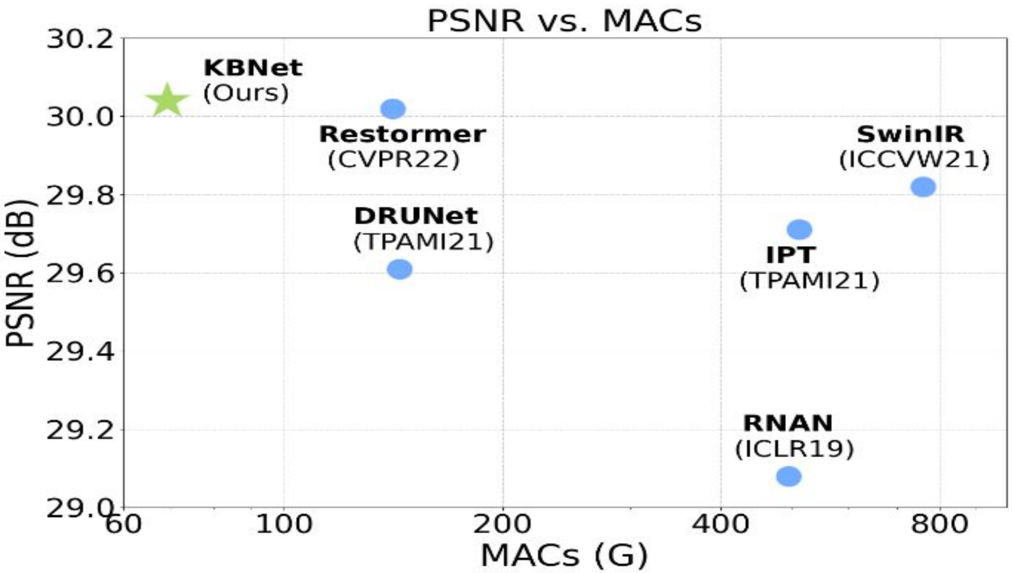
성능 큰 변화 O

2. KBNet

손정우의 한국 최초 KBNet 논문 리뷰

시작하겠습니다 👏🙌

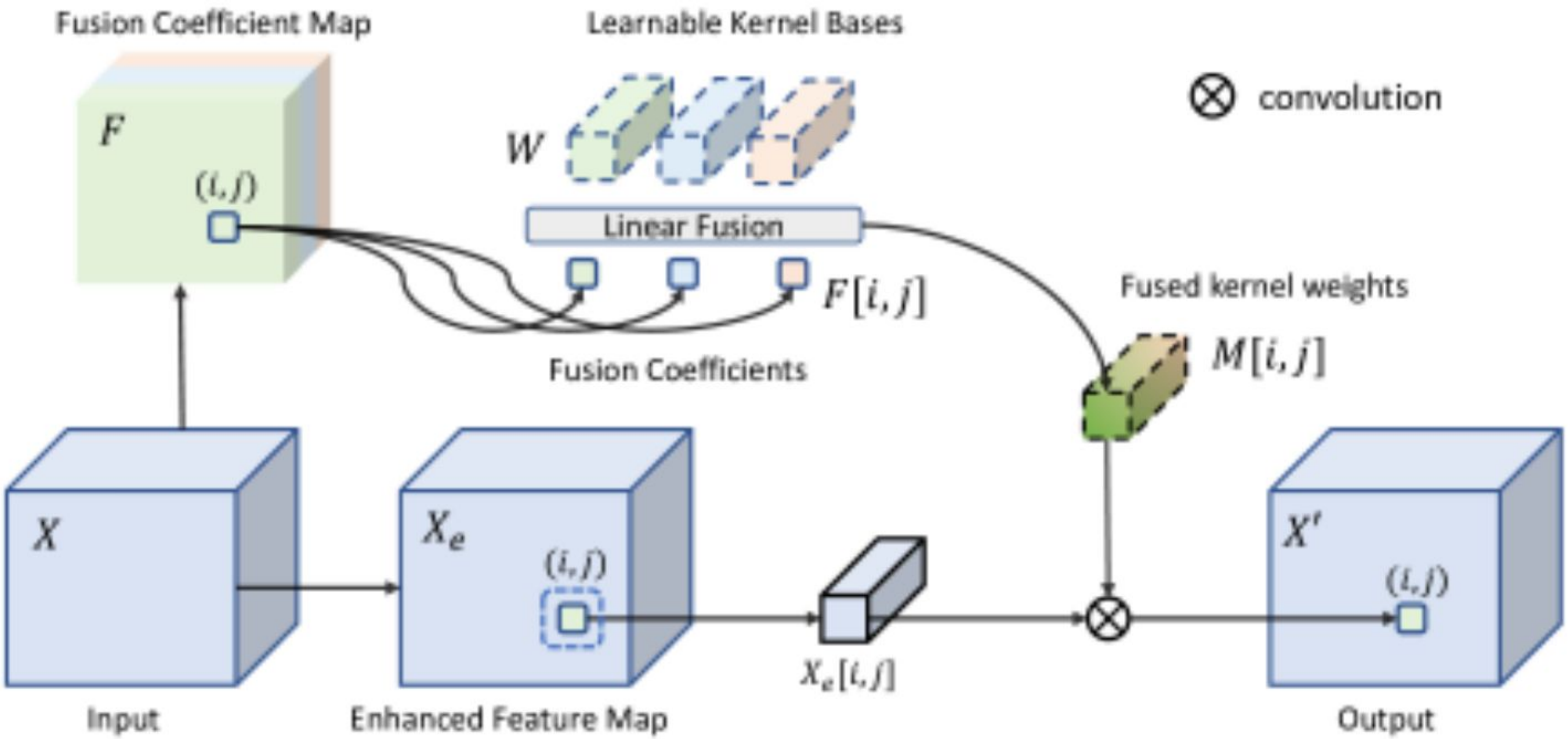
기본 모델 선택 : KBNet



- SIDD라는 데이터셋과 유사한 도메인
- 2023년에 나온 새로운 이미지 디노이징 모델이라 최신 트렌드를 반영할 거라 생각

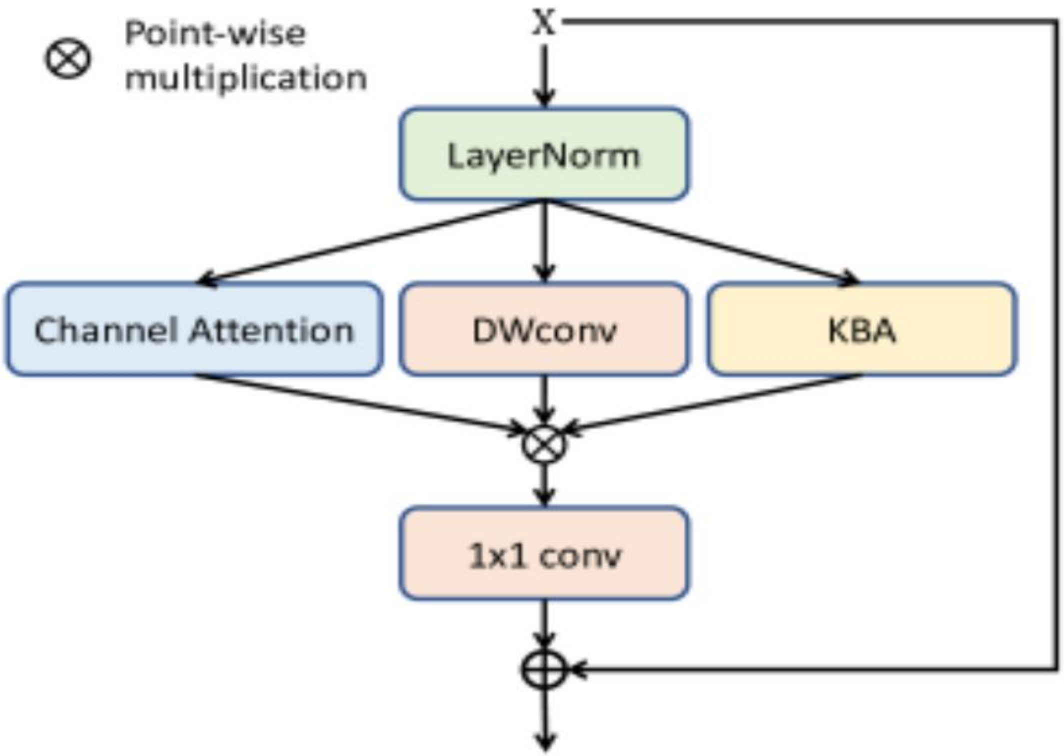
경희대학교
KYUNG HEE UNIVERSITY

KBNet - KBA module



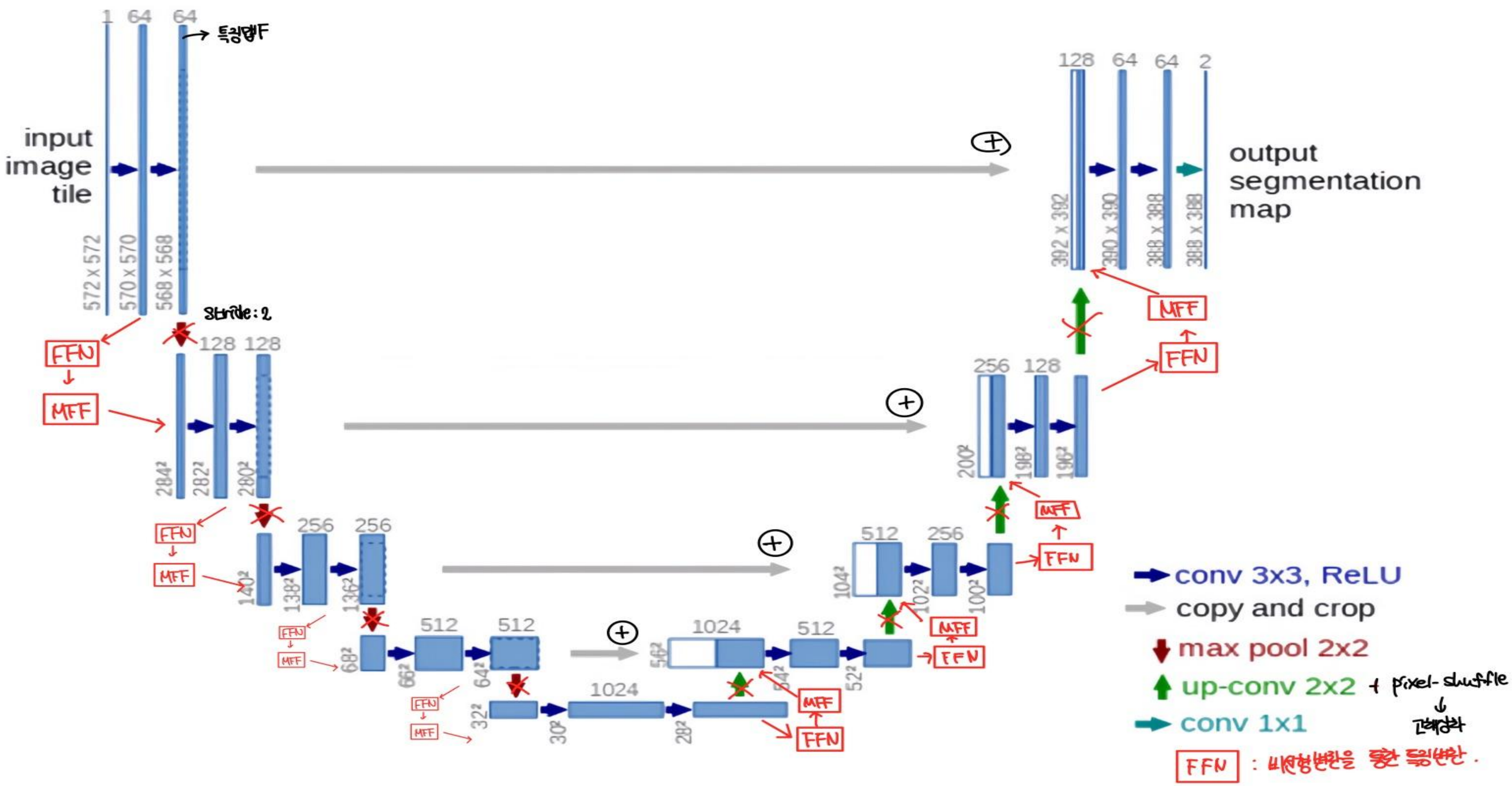
Kernel Basis Attention Module

KBNet - MFF Block

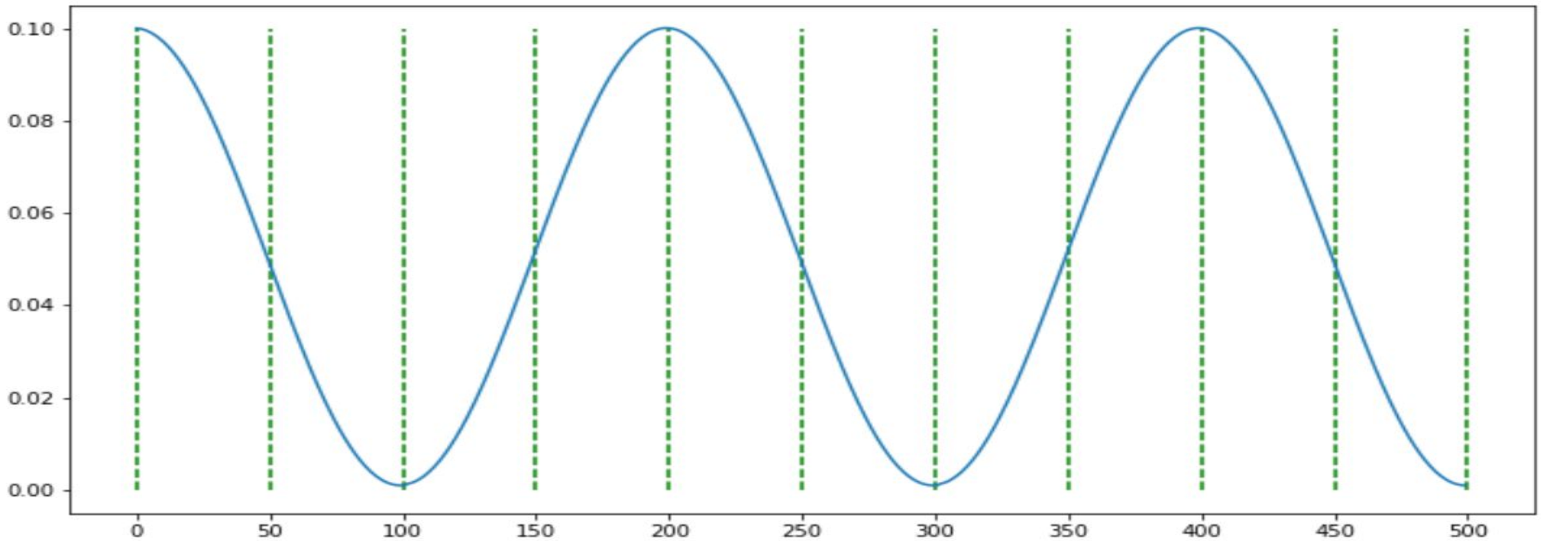


Multi-axis Feature Fusion (MFF) Block

KBNet - model structure



1차 시도



- CosineAnnealingLR 사용
- T_max = 60, eta_min= 0.0001

모델 구조 및 학습 과정

학습 결과

Train Score : 0.0050

Val Score : 0.0052

kaggle score : 3.8

실패 요인 분석

1. Augmentation을 안함
2. Width의 영향으로 파라미터 개수가 적음
3. 학습 시간이 오래걸린 이유 중 하나: batch 크기
4. cos의 최저점 지나간 이후로 성능 개선 X

2차 시도

```python
channel = 3
class CustomDataset(data.Dataset):
    def __init__(self, scan_dir, clean_dir, patch_size = 128, transform=None):
        self.scan_dir = scan_dir
        self.clean_dir = clean_dir
        self.patch_size = patch_size
        self.transform = transform

        self.scan_paths = sorted(os.listdir(scan_dir))
        self.clean_paths = sorted(os.listdir(clean_dir))

    def __len__(self):
        return len(self.scan_paths)

    def __getitem__(self, index):
        scan_path = os.path.join(self.scan_dir, self.scan_paths[index])
        clean_path = os.path.join(self.clean_dir, self.clean_paths[index])

        # Load the clean and scan images using memory-mapped files
        scan_img = self.load_image(scan_path)
        clean_img = self.load_image(clean_path)

        # Apply data augmentation
        scan_img, clean_img = self.augment_image(scan_img, clean_img)

        # Convert Y channel images to PIL Images
        scan_img = Image.fromarray(scan_img)
        clean_img = Image.fromarray(clean_img)

        # Apply data transformation if provided
        if self.transform is not None:
            scan_img = self.transform(scan_img)
            clean_img = self.transform(clean_img)

        return scan_img, clean_img

    def load_image(self, image_path):
        # Open the image file using memory-mapped files
        with open(image_path, "rb") as f:
            mmapped_file = mmap.mmap(f.fileno(), length=0, access=mmap.ACCESS_READ)
            img_data = mmapped_file.read()

        # Read the image data using OpenCV
        img_array = np.frombuffer(img_data, dtype=np.uint8)
```

```python
        elif channel == 3:

            return img

    def augment_image(self, scan_img, clean_img):
        # Random crop
        # Y Channel
        if channel == 1:
            rows, cols = clean_img.shape
            x = np.random.randint(0, cols - self.patch_size)
            y = np.random.randint(0, rows - self.patch_size)
            scan_img = scan_img[y:y+self.patch_size, x:x+self.patch_size]
            clean_img = clean_img[y:y+self.patch_size, x:x+self.patch_size]
            rows, cols = clean_img.shape
        # RGB
        elif channel == 3:
            rows, cols, _ = clean_img.shape
            x = np.random.randint(0, cols - self.patch_size)
            y = np.random.randint(0, rows - self.patch_size)
            scan_img = scan_img[y:y+self.patch_size, x:x+self.patch_size, :]
            clean_img = clean_img[y:y+self.patch_size, x:x+self.patch_size, :]
            rows, cols, _ = clean_img.shape

        # Random rotation
        angles = [0, 90, 180, 270, 360]
        angle = random.choice(angles)
        rotation_matrix = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, 1)
        scan_img = cv2.warpAffine(scan_img, rotation_matrix, (cols, rows))
        clean_img = cv2.warpAffine(clean_img, rotation_matrix, (cols, rows))

        # Random horizontal flip
        if np.random.rand() < 0.5:
            scan_img = np.fliplr(scan_img)
            clean_img = np.fliplr(clean_img)

        # Random vertical flip
        if np.random.rand() < 0.5:
            scan_img = np.flipud(scan_img)
            clean_img = np.flipud(clean_img)

        return scan_img, clean_img
```

- RandomCrop 적용

경희대학교 KYUNG HEE UNIVERSITY

2차 시도

Train Score : 0.0052

Val Score : 0.0049

학습의 성능이 정말 미세하게 좋아짐을 확인하였지만, 학습률이 최저점을 찍고 다시 올라갈 때부터 학습 성능 변화 없음

3차 시도

- 학습 속도를 높이고자 batch 사이즈를 높히려 했으나, 용량문제로 실패
- patch_size 늘이려 했으나, 메모리 문제로 실패
- MAE로 loss 바꿈, 실패
- 스케쥴러를 expotential로 바꿈, 실패
- 초기 학습률 0.005 설정, 4epoch 발산으로 인한 실패
- middle_num개수 늘림, 하지만 계속 enc,dec의 개수 바꾸느라 이로 인한 성능 확인 어려움
- weight-light = true, 로 오버피팅 막고자 함 >> 실패
- weight_penalty를 통해 오버피팅 막고자함, 실패

경희대학교
KYUNG HEE UNIVERSITY

3차 시도

```
class KBNet_s(nn.Module):
    def __init__(self, img_channel=3, width=16, middle_blk_num=4,
enc_blk_nums=[2,2,2,2],dec_blk_nums=[4,2,2,2],
basicblock='KBBlock_s', lightweight=False, ffn_scale=2):
```

-width: 64>>16, enc blk nums = [2,2,2,2], dec blk nums = [4,2,2,2]

>>목적 : 표현력 증가 // 하지만, 실패

4차 시도

width: 64>>16, enc blk nums = [2,3,4], dec blk nums = [4,3,2]

- **모델 용량 축소**

```
if epoch >= 59:
    optimizer.param_groups[0]['lr'] = 0.0002
```

- **학습률을 발산하기 전 epoch에서 고정을 해 발산을 막고자 함**

최종 성능

Train Score : 0.0043

Val Score : 0.0045

kaggle score : 3.47

휴리스틱

1. Mixed Precision
2. DSD

휴리스틱

1. Mixed Precision

가중치 연산에서 소수점 아래 몇 자리에서 잘라내어 학습 속도 향상을 기대.

정확한 수치에서 어느 정도의 오차가 발생하는 셈이니 학습 성능에는 악영향.

IEEE 754 Standard for Floating-Point Arithmetic

1bit	8bit	23bit

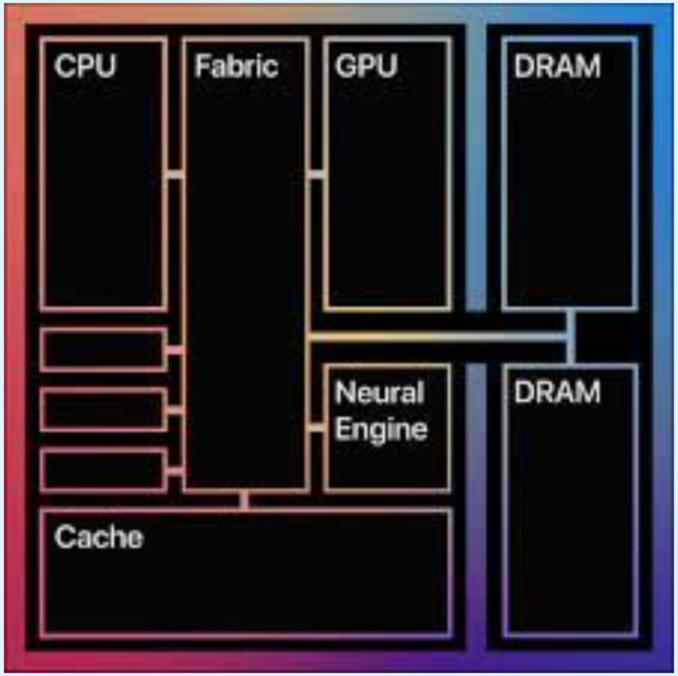
부호(sign) 비트

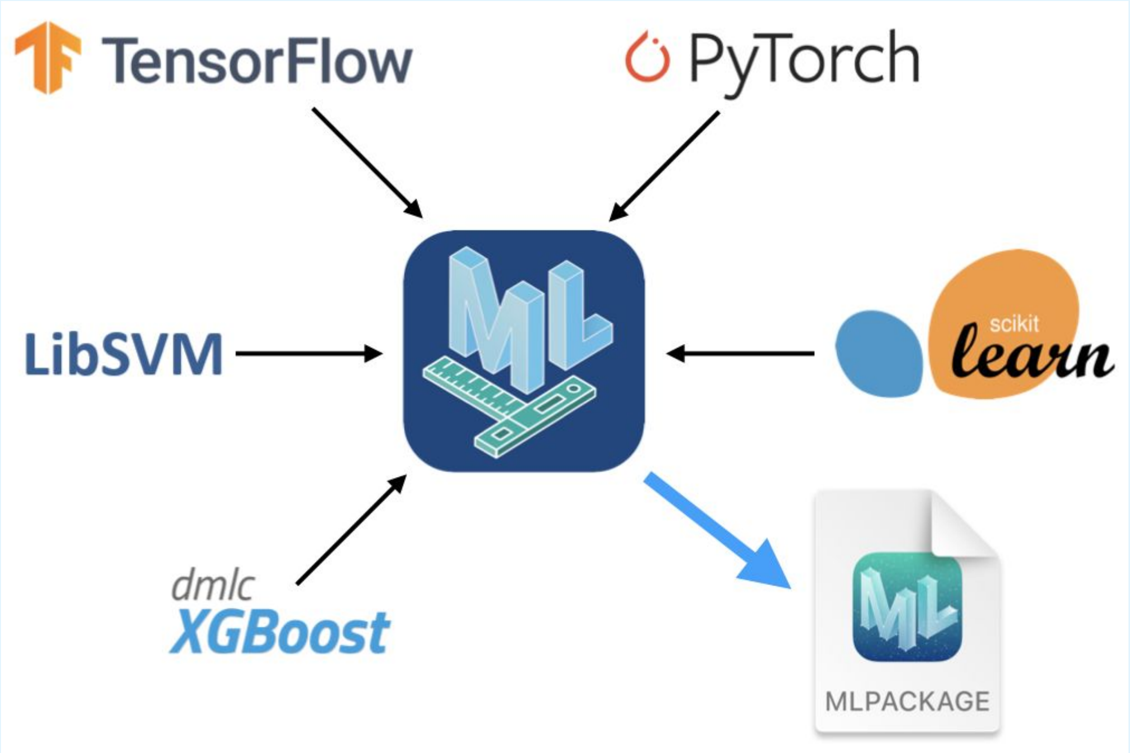
지수(exponent) 비트

가수(fraction, significant, mantissa) 비트

휴리스틱

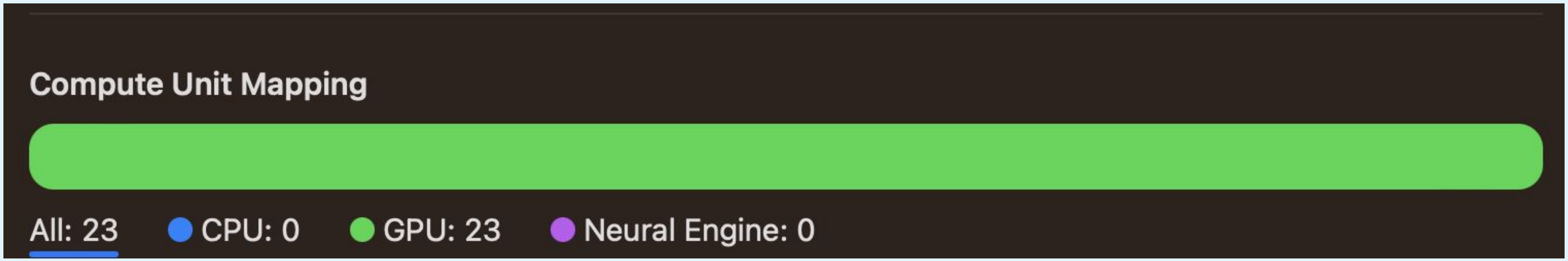
경희대학교
KYUNG HEE UNIVERSITY

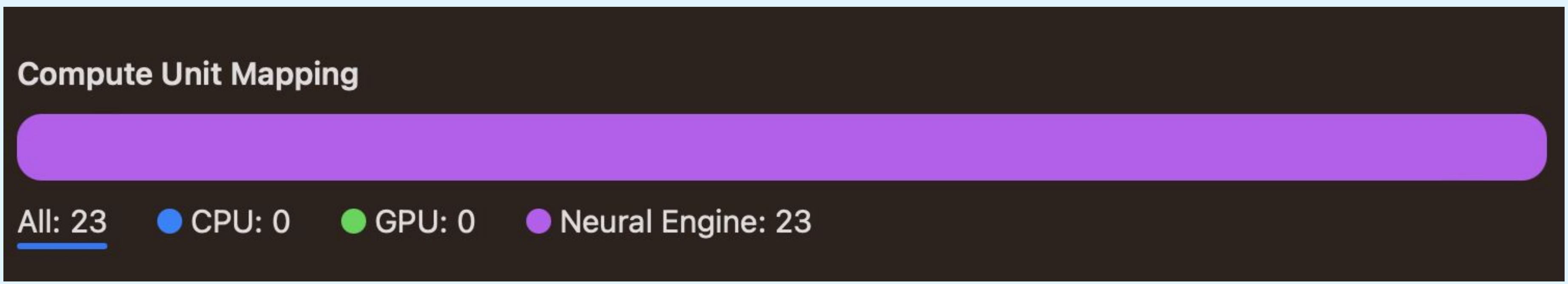




Apple M1 SoC

휴리스틱



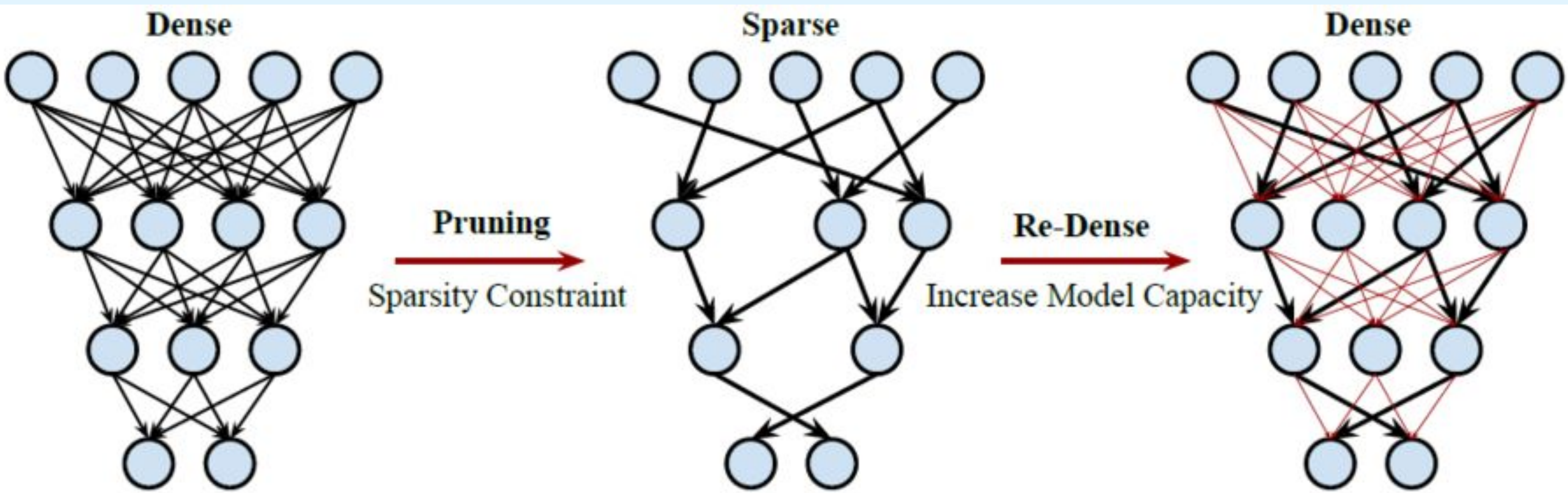


up to 5x faster performance

휴리스틱

2. DSD (Dense Sparse Dense)

Dense 밀집 연결과 Sparse 희소 연결을 번갈아가며 쌓아 불필요한 연결을 줄이고 모델의 계산 비용도 줄여가며 효과적인 정보 전달이 가능함.



감사합니다

2022105780 손정우

2019102212 이정호

2022105773 김재욱

경희대학교
KYUNG HEE UNIVERSITY