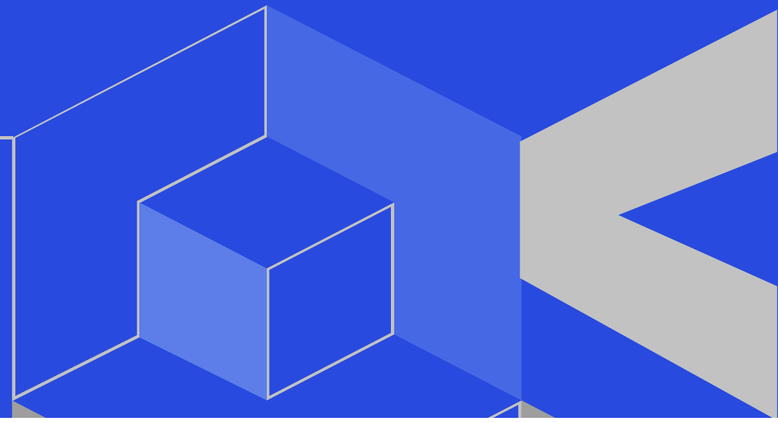
빅웨이브 인턴

중간발표

결초보은 | 김웅기

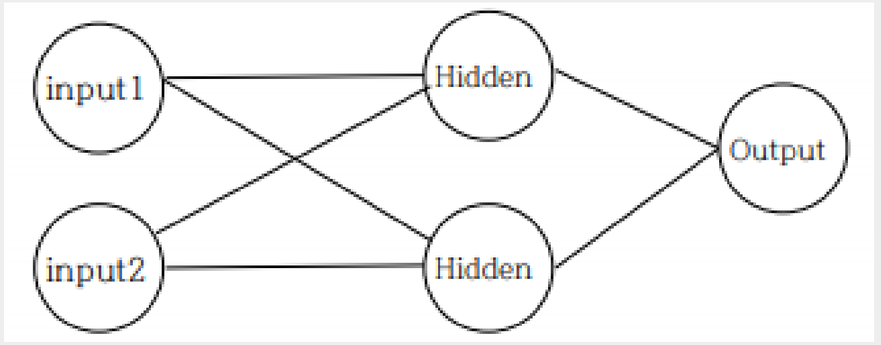


Index.

코딩애플

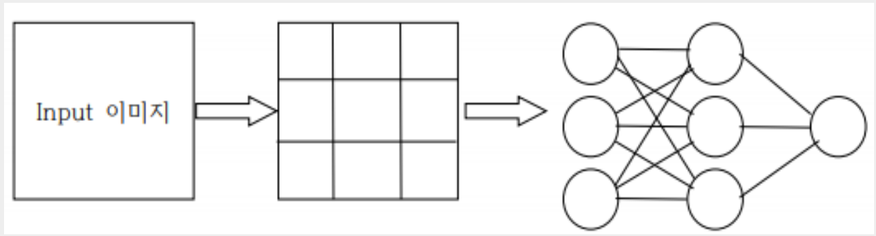
딥러닝 이론

딥러닝 이론



딥러닝 : 머신러닝에서 Input과 Output 사이에
Hidden Layer를 추가한 것

이미지 딥러닝



Input 이미지를 잘게 쪼갠 후 행렬형태로 변환하여 딥러닝 진행

코딩애플

Activation Function

활성화함수

Hidden Layer에 들어가는 함수.
비선형적이고 복잡한 예측을 가능하게 한다는 장점을 가진다.

- Sigmoid

  분류문제에 적용하는 활성화 함수. 특정 기준 이하면 0, 이상이면 1을 출력.
  기울기 손실 문제가 발생한다는 단점을 가진다.

- ReLu (Rectified Linear Units)

  값이 음수면 0, 양수면 값을 그대로 출력.
  기울기 손실 문제가 없지만 값이 음수일 경우 모두 0으로 변환되어 가중치 업데이트가 안될 수 있다는 단점을 가진다.

- Hyperbolic Tangent

  -1 에서 1 사이의 값을 출력하며 평균이 0인 활성화 함수.
  Sigmoid에서 발생하는 기울기 소실 증상이 비교적 적게 발생한다.

코딩애플

Optimizer

Optimizer

손실을 최소화하는 Loss Function을 찾고 최적화하는 파라미터

- Momentum
  가속도를 유지하며 최적화
- AdaGrad
  자주 변하지 않는 W는 크게, 자주 변하는 W는 작게하여 최적화
- RMSProp
  AdaGrad에 제곱을 취한 최적화
- AdaDelta
  AdaGrad에서 a가 너무 작게되어 학습이 안되는 현상을 방지하는 최적화
- Adam
  RMSProp과 Momentum을 결합한 최적화 방법

코딩애플

tensorflow로 모델 구축하기

1. 모델 만들기

- tf.keras.models.Sequential([레이어1, 레이어2, 레이어3])
- tf.keras.layers.Dense(노드수) : 노드 수에는 제한이 없으나 보통 2의 제곱수 기입
- tf.keras.layers.Dense(노드수, Activation=" ") : 마지막 레이어. 출력하고자 하는 값만큼 노드 수를 지정하고 활성화함수 지정

2. 모델 컴파일하기

- model.compile(optimizer = " ", loss = " ", metrics = [])

3. 학습하기

- model.fit(X, Y, epochs =)

4. 예측하기

- model.predict([])

코딩애플

Convolution Layer

CNN

인간의 시신경을 모방하여 만든 딥러닝

Convolution Layer 과정

1. 이미지에서 중요한 정보를 추려서(압축(필터링)) 복사본을 여러 장 만든다.

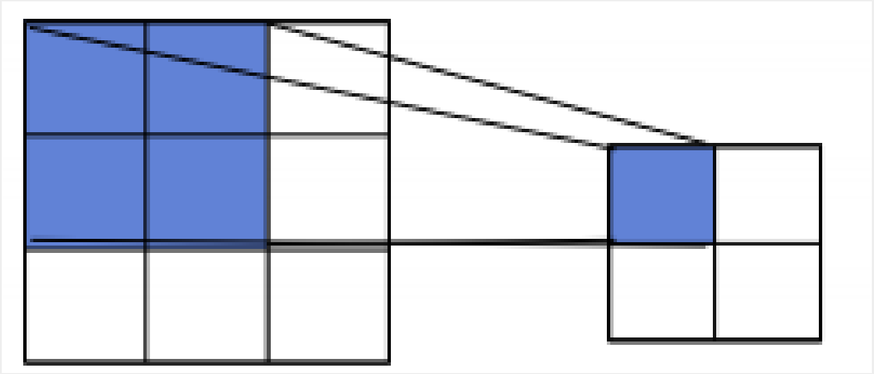
2. 그곳엔 이미지의 중요한 Feature 특성이 담겨저 있다.
(이미지 특성이 각각 다르게 강조되게)

3. 1, 2를 통해 특성 추출(Feature Extraction)

코딩애플

Pooling

Feature map



단순 Convolution의 문제점 : Feature의 위치
-> 해결책 : Pooling Layer

- **Max Pooling(최대값만 추림)**

  Layer1에서 색칠된 부분의 Feature 중 최대값을
  Layer2에 전달하는 방법

- **Average Pooling(평균값으로 추림)**

  Layer1에서 색칠된 부분의 Feature 의 평균을
  Layer2에 전달하는 방법

코딩애플

Segmentation

Image Segmentation

증강된 이미지의 사본을 생성.
원본 이미지 + 증강 이미지를 사용하여 모델을 학습한다.

- 이미지 뒤집기

  layers.experimental.preprocessing.RandomFlip

- 이미지 위치 랜덤 지정

  layers.experimental.preprocessing.RandomRotation

- 이미지 확대

  layers.experimental.preprocessing.RandomZoom

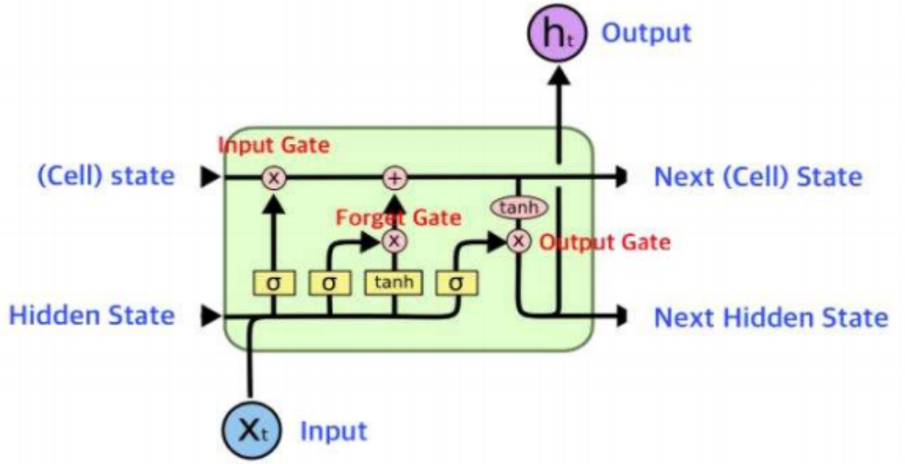
원본 데이터가 적거나 너무 많을 경우 효과가 없다.
시간이 오래 걸린다.

코딩애플

LSTM

LSTM

RNN에 장기기억층(cell state)를 추가한 모델



- 장기기억층에 이전 단어 혹은 데이터의 가중치를 보존해두고 모델을 학습하는 방식

- tensorflow에서 LSTM 만들기
  tf.keras.layers.LSTM(노드수, input_shape=(,))

코딩애플

(Generative Adversarial Net)

GAN

적대적 신경망.
모델 2개를 경쟁시켜 새로운 데이터를 만드는 모델
Generator 모델(가짜 데이터 생성)
+
Discriminator 모델(생성 데이터 검증)

기본적인 GAN 학습 Step

- Discrimanator

  1. 실제 이미지와 가짜 이미지 준비
  2. 구분 잘하게 학습 (model.train_on_batch(이미지, 정답))

- Generator

  1. 랜덤 숫자 행렬 100개 준비
  2. 1로 라벨링해서 집어넣기

- 모델 합치기

  Sequential([Generator 모델, Discriminator 모델])

object detection

You Only Look Once

YOLO

- Object Detection(객체 인식)

1. 이것은 무엇인가? (특정 이미지에서 대상을 식별)
2. 어디에 위치해있는가? (이미지 내에서 대체의 정확한 위치 식별)

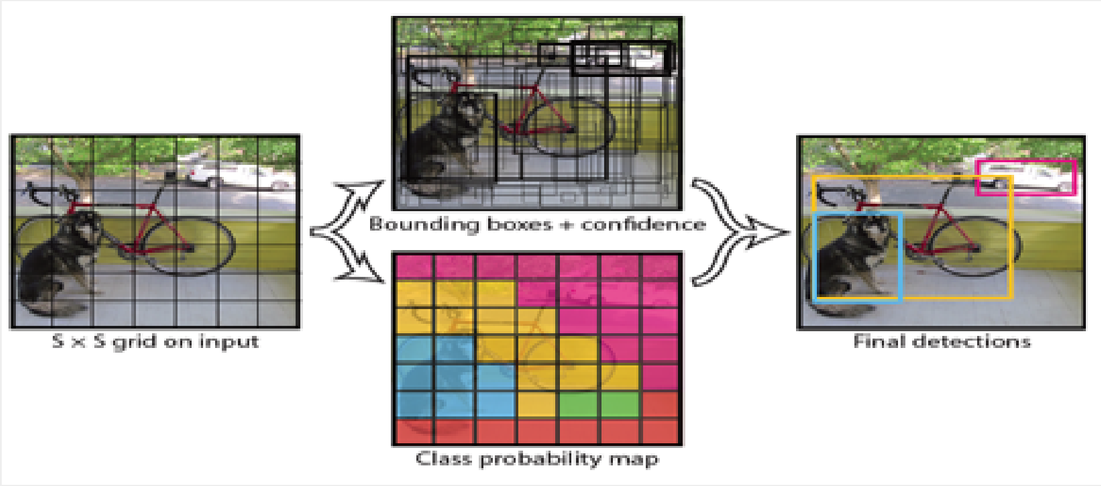
- YOLO(You Only Look Once)

1. 이미지 입력
2. 입력된 이미지를 일정 분할로 그리드
3. 신경망 통과
4. 바운딩 박스와 클래스 예측
5. 최종 감지 출력

object detection

You Only Look Once

YOLO



분할한 그리드별 바운딩 박스와 클래스를 예측
NMS 알고리즘을 적용해 최종적으로 남길 바운딩 박스 외에 박스들을 제거

실습한 코드에서 이미지 예측 결과 출력하기

```
!python detect.py
--weights /content/yolov5/runs/train/gun_yolov5s_results2/weights/best.pt
--img 416 --conf 0.5 --source "{val_img_path}"
```

- img : 출력 이미지 크기 설정
- conf : 남길 바운딩 박스의 점수 설정

Image Augmentation

Augmentation

이미지 학습마다 이미지를 변형시켜 학습하는 방법

- tf.keras.layers.UpSampling2D(size=)
  저해상도 이미지를 고해상도 이미지로 변경
- tf.keras.layers.Conv2DTranspose(kernels = , filter_size =)
  Convolution연산이 들어가서 해상도를 키움.
  학습과정에서 필터가 학습이 됨
- ReduceLROnPlateau
  콜백함수.
  모델 개선이 없을 경우 Learning Rate를 조절해 모델의 개선을 유도

```
model.fit(callbacks = [ReduceLROnPlateau
(monitor='val_loss', factor=0.2, patience=10, verbose=1,
mode='auto', min_lr=1e-05)])
```

SAM

Segmentation Anything Model

SAM

META에서 발표한 VIT-Huge 기반 이미지 특성 추출 모델

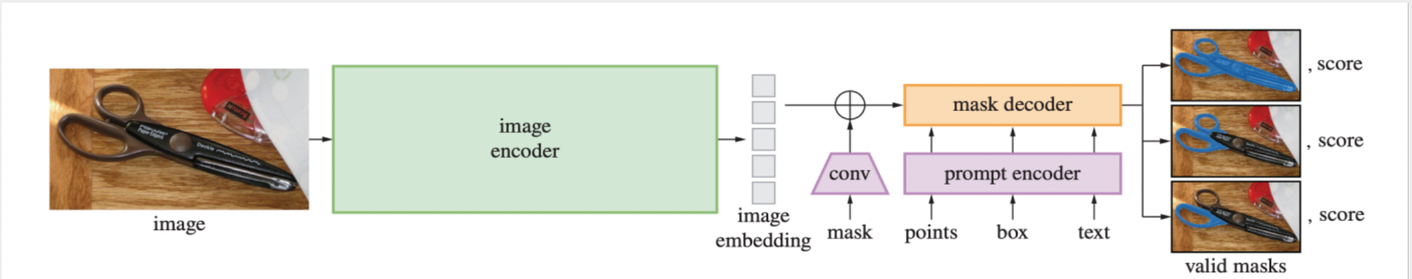


Figure 4: Segment Anything Model (SAM) overview. A heavyweight image encoder outputs an image embedding that can then be efficiently queried by a variety of input prompts to produce object masks at amortized real-time speed. For ambiguous prompts corresponding to more than one object, SAM can output multiple valid masks and associated confidence scores.

1. Input 이미지를 Encoder에 입력
2. image embedding
   (NLP에서 사용하던 Transformer를 사용한다는 특성)
3. pixel별 마스크를 씌우고 conv레이어를 통해 특성 추출
4. 마스크별 point 생성 후 bounding box 생성
   (여기서 여러 개의 point가 묶여 하나의 객체로 생성됨)
5. 객체별 점수 예측

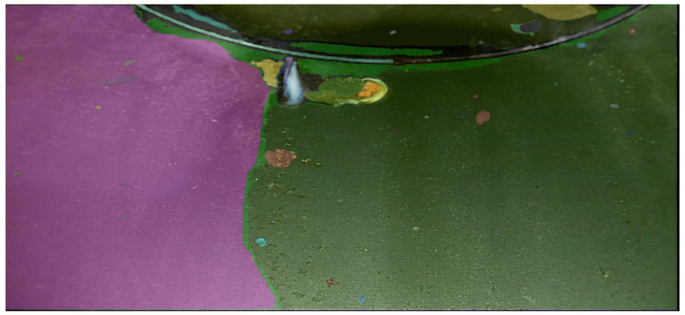
SAM

Segmentation Anything Model

SAM

파라미터

- points_per_side : 사용할 포인트 수
- pred_iou_thresh : 예측한 바운딩 박스의 최소 iou값 (이하면 박스 삭제)
- stability_score_thresh : 예측한 클래스의 최소 점수 (이하면 박스 삭제)



감사합니다.

결초보은 I 김웅기