ALOC

2025. 06. 24

최종발표

Deep하게 Deep Study

3기 이도권, 2기 박도현, 2기 유영호, 3기 김동현, 3기 김준수, 3기 이준형

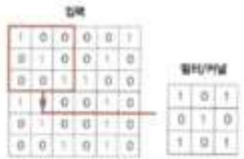
ALOC

PyTorch

- 주요 인공지능 기술 공부.
- **"PyTorch"**를 통해 실습.
- 매주 정해진 분량을 개인 공부 진행.
- **1**팀/**2**팀은 각 주차 해당 파트에 대하여 발표를 진행.
  (1팀**)** 이도권 / 김준수 / 이준형
  (2팀**)** 유영호 / 박도현 / 김동현
- 추후 프로젝트를 위한 스터디.

ALOC

5주차 스터디 학습 확인 프린트 (Deep하게 Deep스터디)



* 위는 입력된 이미지

Q. stride(커널이동범위)가 1일 때, feature map은?

Q. stride(커널이동범위)가 3일 때, feature map은?

1	2	3	4	5
2	1	0	1	2
3	0	1	1	0
1	4	1	1	2
2	1	1	0	0

Q. 위 feature map에 패딩(padding)을 한 결과는? (패딩 값으로는 0을 사용한다.)

(패딩을 하는 이유는 보통 얻어지는 feature map은 입력보다 작아지기 때문)

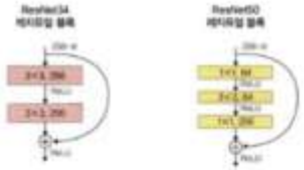
6주차 스터디 학습 확인 프린트 (Deep하게 Deep스터디)

```
# zip()은 여러 개의 리스트(or tuple)를 합쳐서 새로운 tuple 타입으로 반환
a = [1, 2, 3, 4]
b = ['soju', 'beer', 'wine', 'tequila']
for x, y in zip(a,b):
        print(x, y)
```

Q. 위 코드의 실행 결과를 적어보시오.

Q.

ResNet에서



위 그림은 병목 블록을 두지 않았을 때와 병목 블록을 두었을 때에 대한 그림이다.

병목 블록을 사용하면 어떤 효과가 생길 수 있는가?

3주차	발표 파트 정하기 / 머신러닝 & 딥러닝 기초	Deep하게 Deep 스...
4주차	PyTorch 기초	스터디 추가 자료(AL...
5주차	퍼셉트론(MLP 포함) & 역전파 알고리즘	Adam Algorithm.p... vanishing_gradien... ConvNeXt.pptx
6주차	CNN	cnn nlp.pptx GNN.pptx
7주차	RNN	seq2seq presentat... QRNN.pptx
8주차	LSTM & Transformer	m2m100.pptx Mamba.pptx transformer 구조.p...

ALOC

머신러닝 & 딥러닝 기초
1차

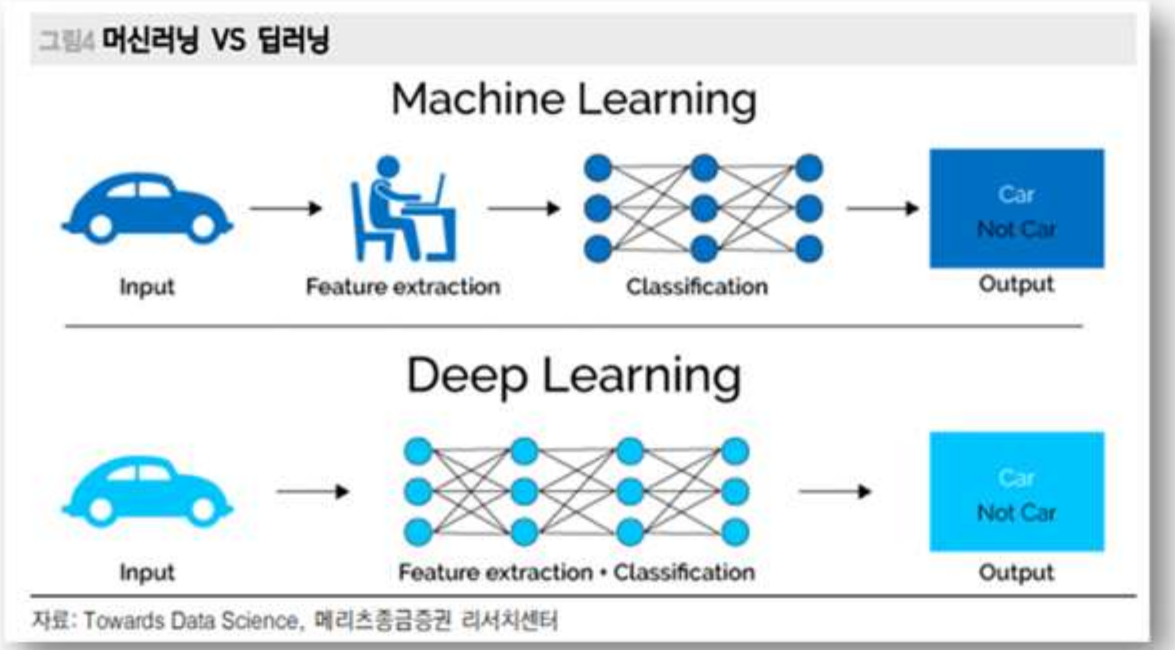
PyTorch
2차

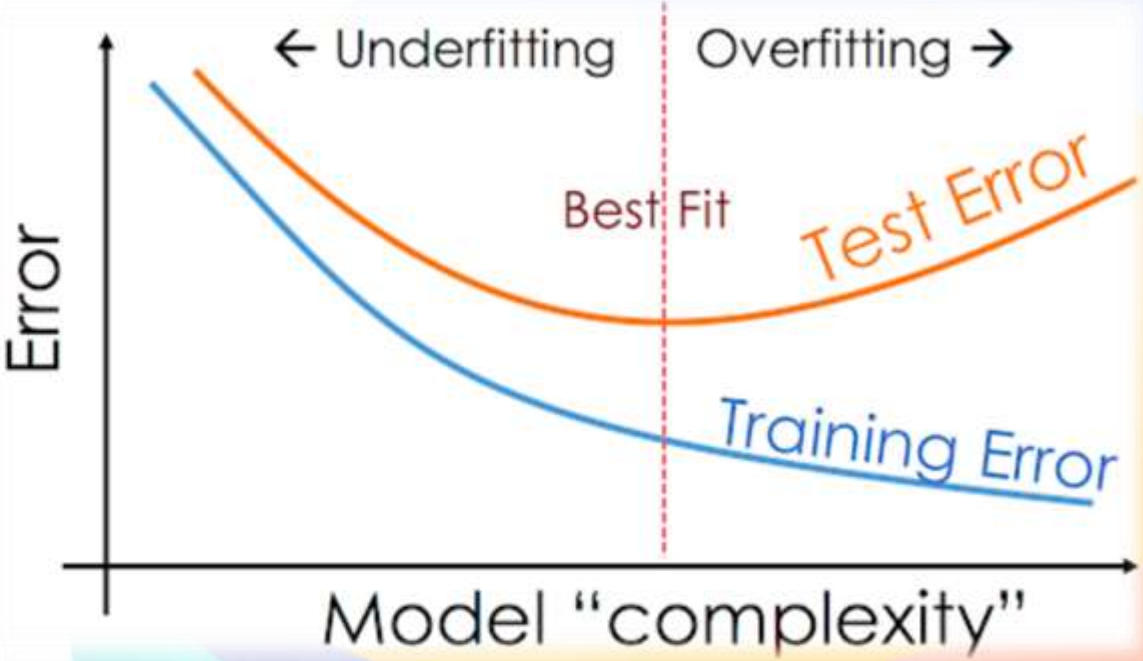
MLP & Backpropagation
3차

DNN & 최적화 기법
4차

RNN & LSTM, Transformer
5차-7차

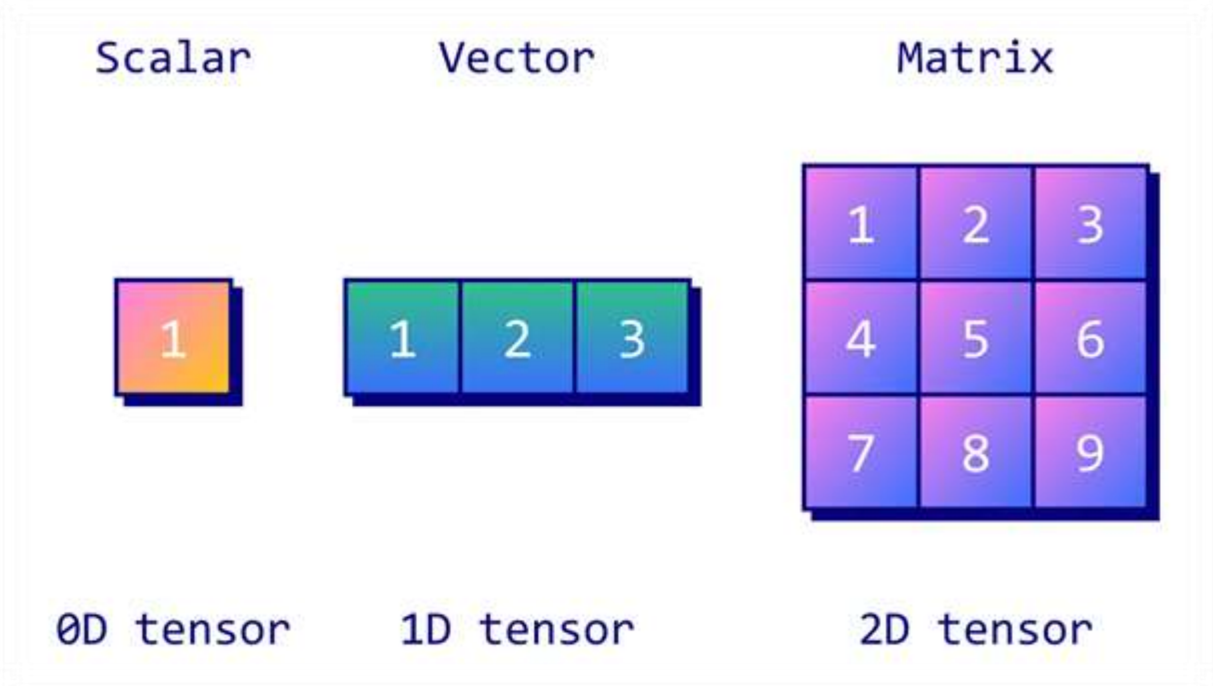
1차 – 머신러닝과 딥러닝 / 과적합, 과소적합





ALOC

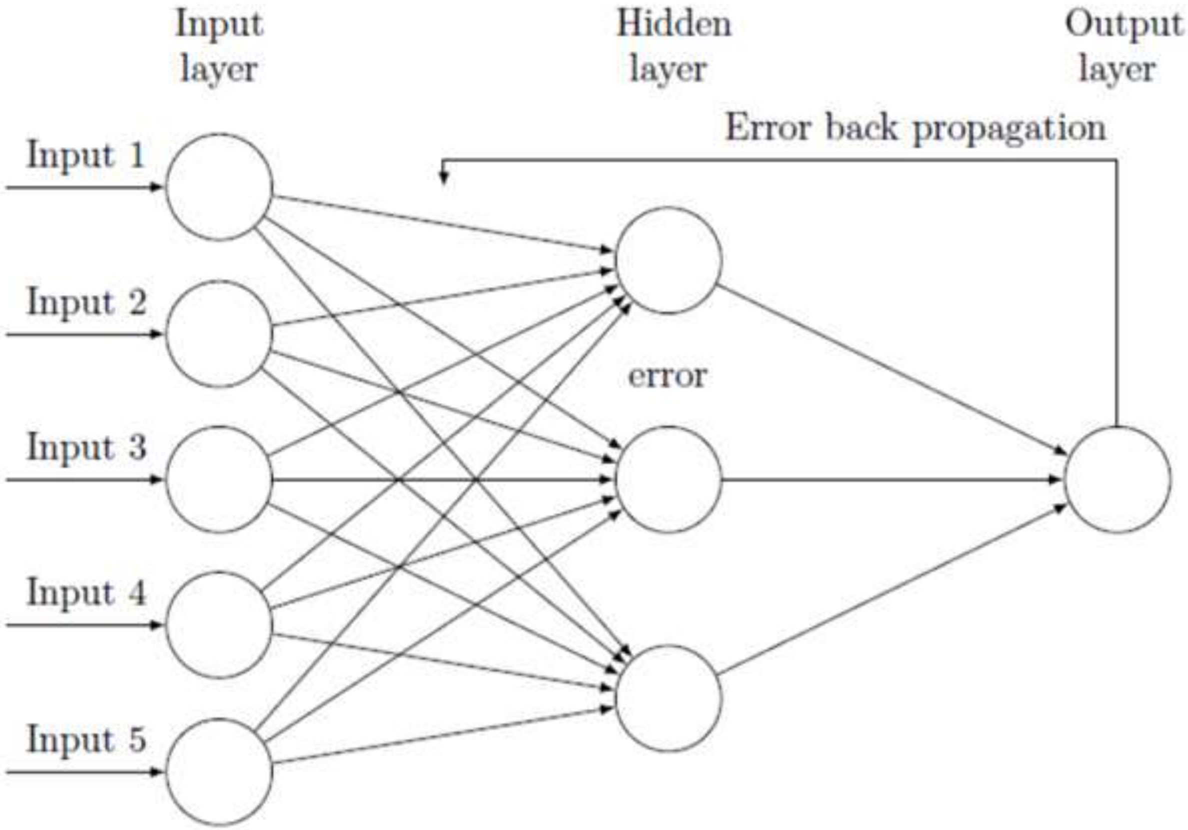
2차 – PyTorch 라이브러리 사용, Tensor의 개념



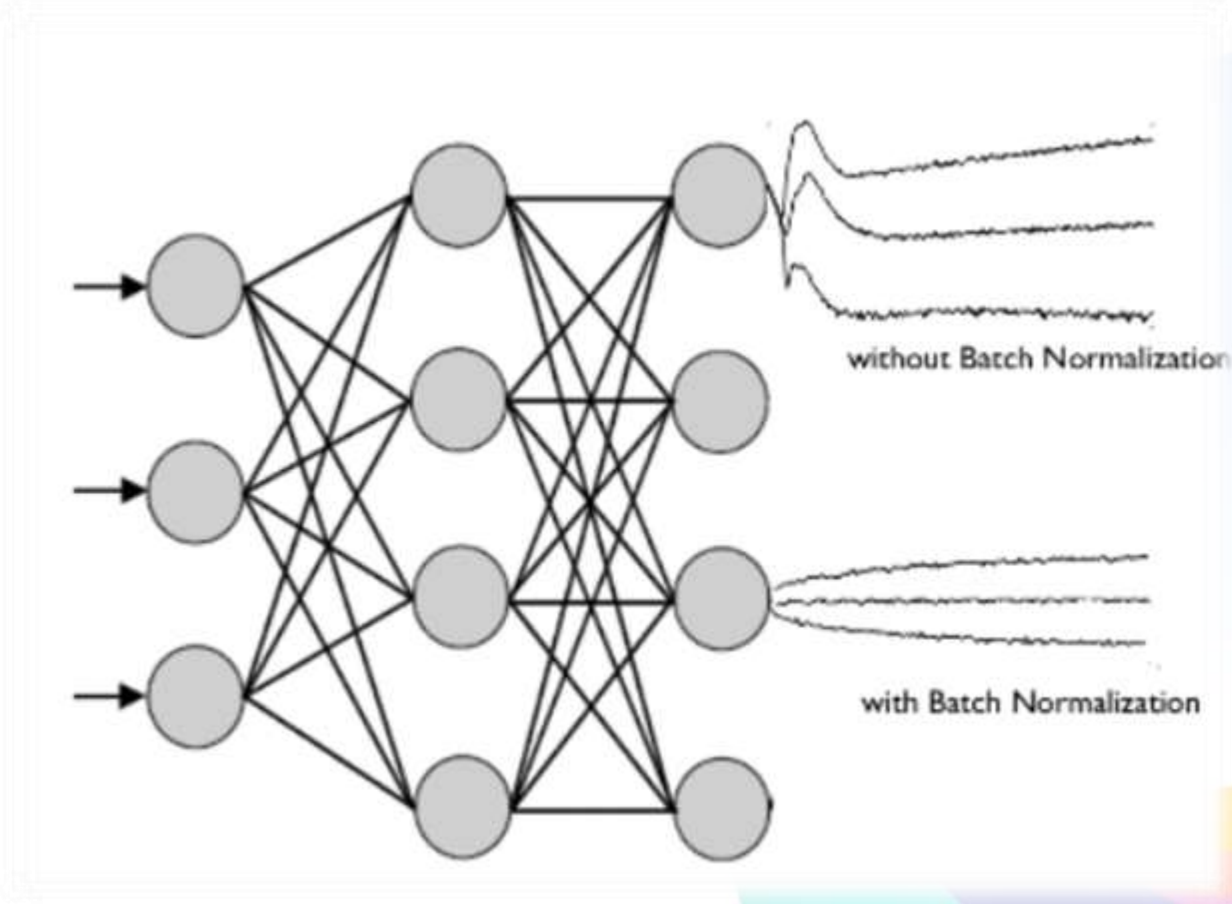

```
mport torch
rom functorch import functionalize
rom torch.fx.experimental.proxy_tensor import make_fx

lef f(x):
    y = x.clone()
    y.add_(1)
    return y

functionalized_f = functionalize(f)
x = torch.ones(4)
print(torch.allclose(f(x), functionalized_f(x)))

# Print an FX graph of the "functionalized" version of f
fx_g = make_fx(functionalized_f)(x)
print(fx_g.code)
```

ALOC

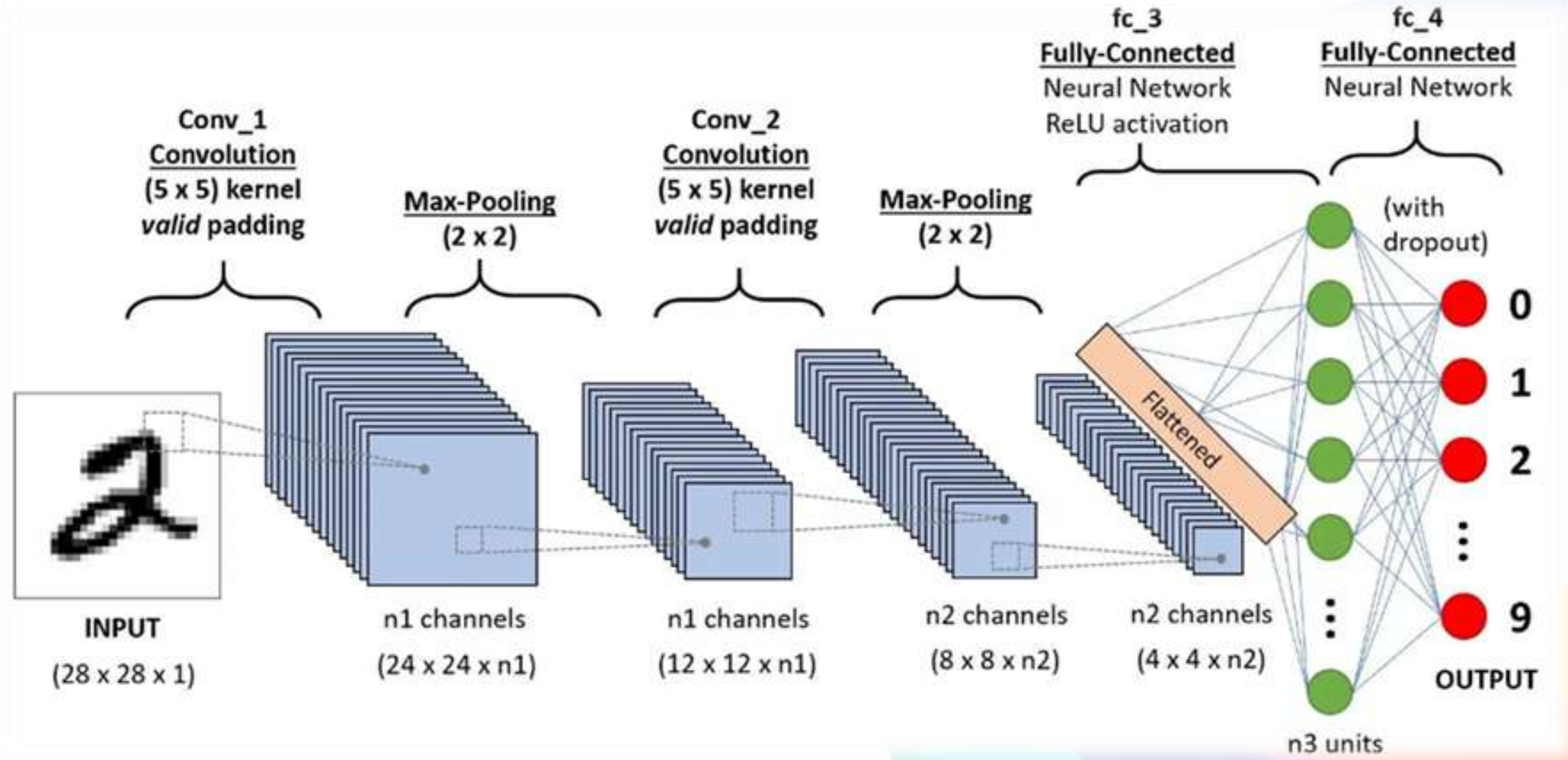
3차 – MLP구조, 역전파 알고리즘



4차 – 심층 신경망 설계, 최적화 알고리즘

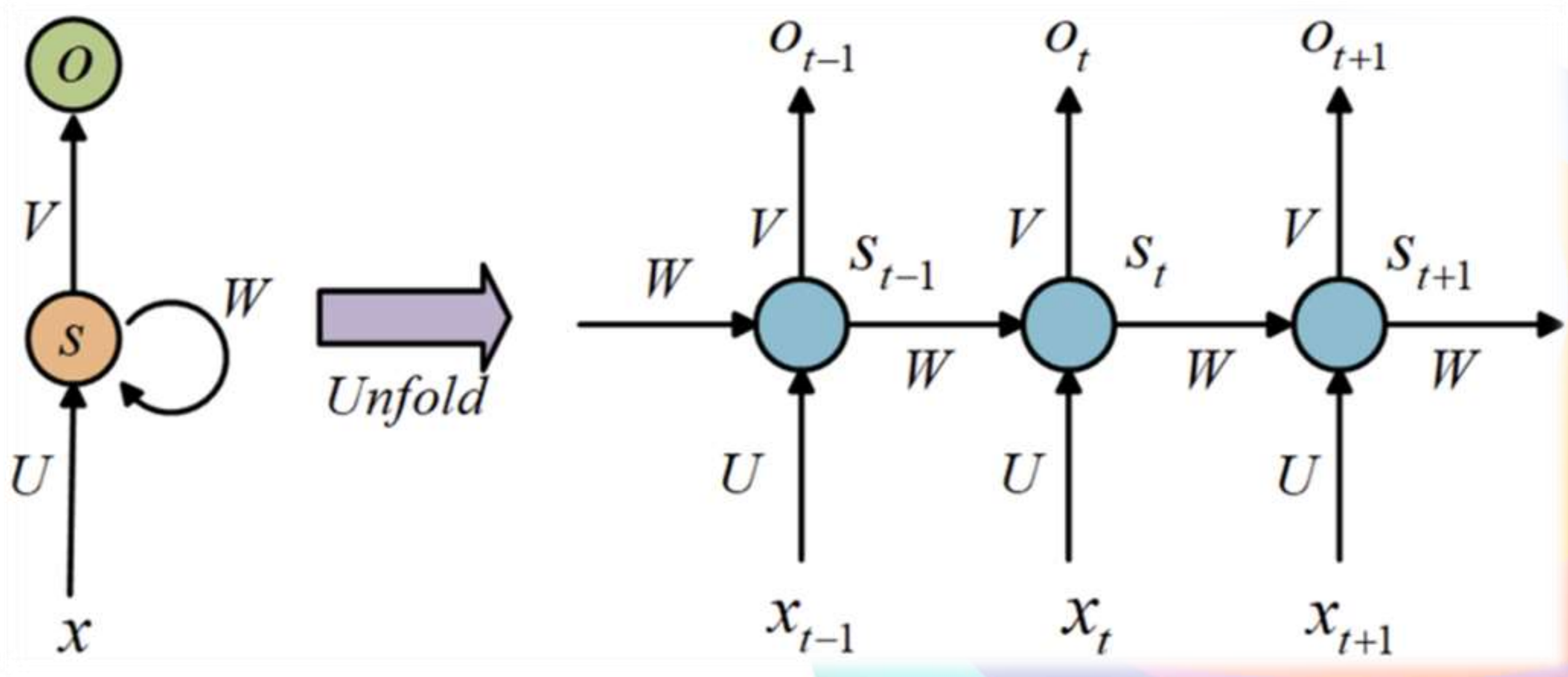


5차 – CNN구조

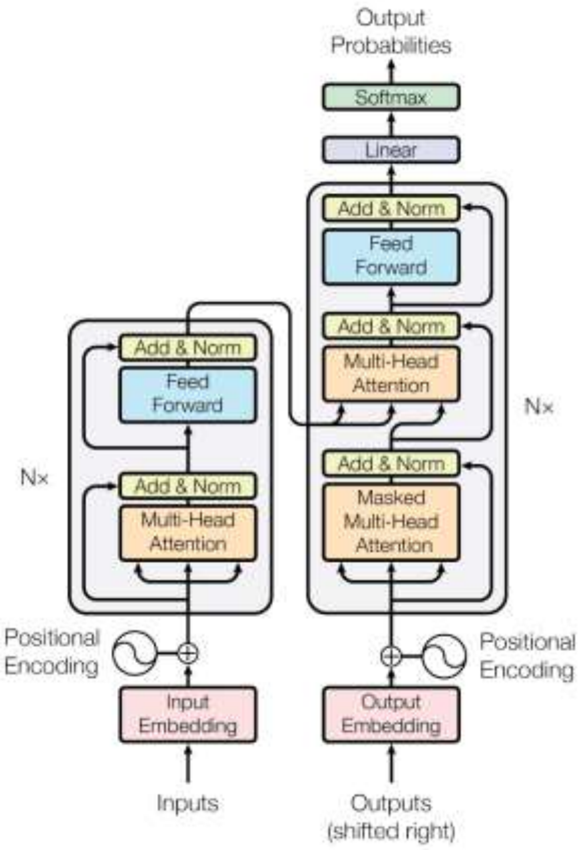


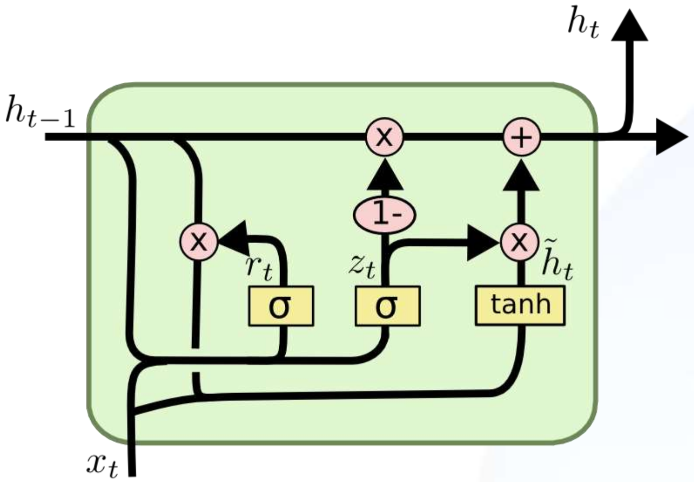
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

6차 - RNN 구조



7차 – LSTM과 Transformer





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

깊이 있는 탐구 사례 _이도권

Adam Optimizer

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla f(x_{t-1})$$

$$g_t = \beta_2 g_{t-1} + (1 - \beta_2)(\nabla f(x_{t-1}))^2$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{\hat{g}_t + \epsilon}} \cdot \hat{m}_t$$

β_1 : Momentum의 지수이동 평균 $\approx$ 0.9

β_2 : RMSProp의 지수이동 평균 $\approx$ 0.999

$\hat{m}, \hat{g}$: 학습 초기 시 m_t, g_t 가 0이 되는 것을 방지하기 위한 보정 값

ϵ : 분모가 0이 되는것을 방지하기 위한 작은 값 $\approx 10^{-8}$

η : 학습률 $\approx$ 0.001

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset

# 훈련 데이터
x_train = torch.randn(100, 10)
y_train = torch.randn(100, 1)

train_dataset = TensorDataset(x_train, y_train)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

# 모델
class SimpleMLP(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x

model = SimpleMLP(input_dim=10, hidden_dim=20, output_dim=1)

# 손실 함수 정의 (MSE 사용)
criterion = nn.MSELoss()

# Adam Optimizer 생성
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 학습 루프
num_epochs = 10

for epoch in range(num_epochs):
    running_loss = 0.0

    for inputs, targets in train_loader:
        # 순전파
        outputs = model(inputs)
        loss = criterion(outputs, targets)

        # 역전파 및 최적화
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_loss += loss.item() * inputs.size(0)

    epoch_loss = running_loss / len(train_loader.dataset)
    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}')
```

활동 중 느낀 점 및 회고 _이도권

깊이 있는 탐구 사례 _이준형

ViT vision Transformer

이미지를 토큰으로 분리, NLP처럼 Transformer 구조로 처리.

"An Image Is Worth 16X16 Words : Transformers for Image Recognition at Scale."

→ 논문 읽고 정리했음.

ViT 전체구조

image를 패치로 분할 후, NLP의 단어처럼 취급하여
각 패치의 Linear Embedding을 순서대로 Transformer의
입력으로 넣어 분류를 수행함.

등장배경

*원래 Transformer의 특징

"The dominant approach is to pre-train on a large text corpus
and then fine-tune on a smaller task-specific dataset).
Thanks to Transformers' computational efficiency and scalability, ..."

↳ self-attention을 활용하여 Sequence model(RNN, LSTM)들에 비해
연산이 효율적이고, 병렬화에 스케일링이 가능함.

⊕ NLP분야에서 Transformer를 활용한 성과가 존재.

때문에

→ 표준 Transformer를 적용한 연구함(최소한 수정(변경)을 통해서.)

*적은 규모의 데이터셋(ImageNet)에서 Transformer가 ResNet보다 성능이 낮음.

CNN만의 고유한 귀납적 편향 (평행이동 불변성, 지역성)이 X
translation equivariance locality

하지만 대규모 (14M-300M images)에서 충분한 데이터 pre-training, fine-tuning 하면
Transformer가 매우 뛰어남.

상세구조

1. 2.

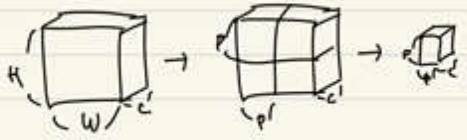
전체 이미지 → 고정크기 패치 → 선형임베딩 & 위치 임베딩 & 학습가능한 분류토큰 추가
→ Transformer Encoder → MLP Head → Result

3.

1. 이미지 $x \in R^{H \times W \times C}$ → P×P 크기의 이미지 $N(=H \times W / P^2)$개로 분할, Sequence $x_p \in R^{N \times (P^2 \cdot C)}$로 reshape

- (H,W): 원본 이미지 해상도
- C: 채널 수
- (P,P): 각 Image 패치 해상도

ViT는 2D Image 처리 → 입력 형태 처리함.



"Encoder 전체 정보를 저장"

2. Embedding

Learnable Embedding 클래스 임베딩

$z_0^0 = x_{class}$

↳ 이 토큰의 Transformer 인코더 최종출력

→ z_L^0는 이미지 전체의 표현 벡터 y로 사용

$y = LN(z_L^0)$

position embedding → 보통 1D. 2D여도 성능 차이 크지 X

+) Learnable class 임베딩과 patch 임베딩에 더함.

추가

Learnable class는 class token을 만들고 다른 token들에 concat함.

「Transformer의 모든 레이어에는 동일한 벡터 차원 'D'를 사용.

∴ 이미지 패치를 평탄화한 뒤, Learnable Linear Projection을 통해 D차원으로 매핑.
학습가능한 선형 변환」

↳ $Z_0 = [x_{class}; x_p^1E; x_p^2E; \cdots; x_p^NE] + E_{pos}$ $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$, $E_{pos} \in \mathbb{R}^{(N+1) \times D}$

3. MLP

구성

Transformer Encoder
- multi-head self-attention (MSA) → $z'_l = MSA(LN(z_{l-1})) + z_{l-1}$ | $l = 1 \cdots L$
- MLP 블록들 → $z_l = MLP(LN(z'_l)) + z'_l$ | $l = 1 \cdots L$
- Layer Norm (LN) 정규화
  (GeLU 활성화 함수 사용)

특징

귀납적 편향

CNN은 많이 반영된 translation equivariance, locality 등이 모델 전체의 모든 계층에 지속적으로 내장

But ViT는 MLP 계층에서 locality 있고, translation equivariance 있으며, Self-Attention 계층은 global한 연산을 수행.

패치 자를 때 활용

+ 2차원 구조를 제한적으로만 사용한다.
  - 이미지 자를 때
  - 이미지 fine-tuning 시 위치 임베딩을 조정할 때
    ↳ 따라서 패치 간 공간적 관계는 모델이 처음부터 학습해야 함.

4. fine-tuning, 고해상도 처리

- 대규모 dataset로 학습 후, 작은 downstream task로 최적을 위해 fine-tuning을 한다.

이 때 사전학습된 예측 헤드를 제거하고, D×K feed forward 계층을 추가한다.
- D는 임베딩 차원
- K는 작은 downstream task의 클래스 수. (이 때 가중치는 0으로 초기화)

- 고해상도 처리 시 pre-training 단계에서 사용했던 patch size와 동일한 size 사용

→ seqgence 더 길어짐.

+ 원본 위치에 따라 다시 2D interpolation (2D 보간) 수행한다.

*이 과정은 Resolution adjustment. 해상도 조정 및 patch 추출 시 inductive bias 수동 주입.

∴ 이 과정이 Vision Transformer에
2D 구조에 대한 inductive bias를 명시적으로
주입하는 유일한 지점.

5. 실험 결론.

ViT → 이미지 인식에 Transformer를 직접적으로 적용한 연구를 하였음.

Self-Attention을 이용한 다른 Computer Vision과 달리 이미지에 대한 귀납적 편향 도입 X.

대신 이미지를 NLP처럼 시퀀스로 나누어 Encoder에 입력시킴.

이러한 단순하면서도 확장 가능한 전략적 방식이 놀라운 결과를 만들어냄

Simple scalable strategic

But detection과 segmentation 과 같은 다른 CV 작업에 적용하기 위해 도전과제 있음.

또한 여전히 self-supervised pre-training과 large-scale self-supervised pre-training 사이

차이가 존재함.

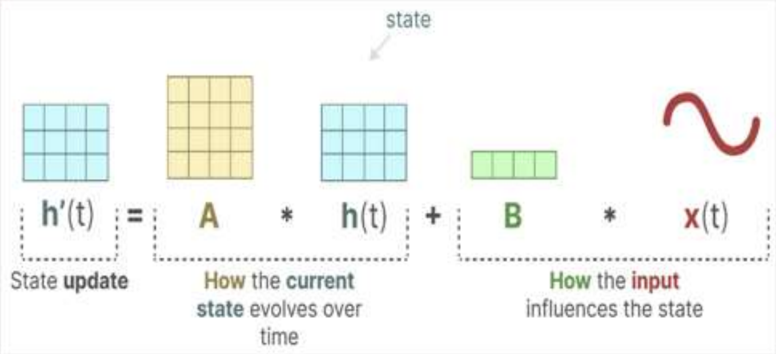
활동 중 느낀 점 및 회고 _이준형

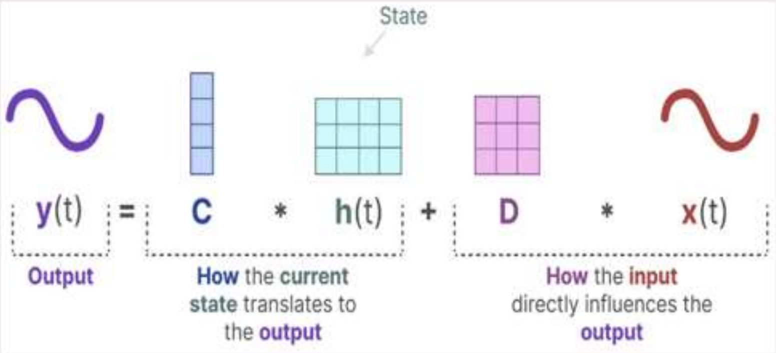
깊이 있는 탐구 사례 _김동현

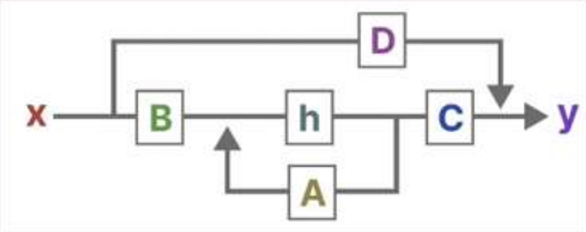
ALOC

구조 요소	Transformer	Mamba
기본 연산	Self-Attention + MLP	Selective SSM
연산 복잡도	$O(n^2)$	$O(n)$
계층 구조	여러 블록 쌓기	여러 Mamba블록 쌓기
하드웨어 효율성	상대적으로 낮음	높음

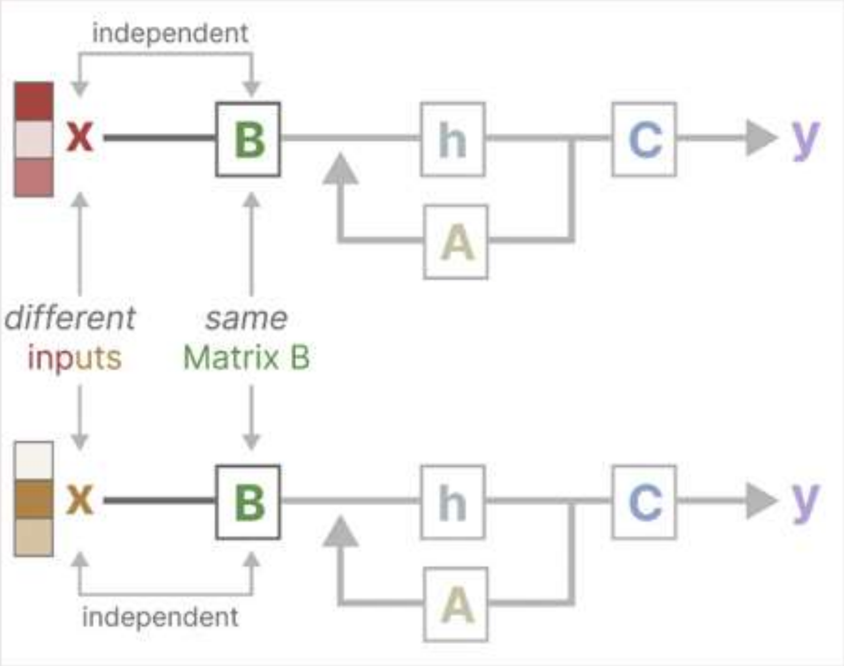
2 | SSM


state
h'(t) = A * h(t) + B * x(t)
State update
How the current state evolves over time
How the input influences the state


State
y(t) = C * h(t) + D * x(t)
Output
How the current state translates to the output
How the input directly influences the output


D
x
B
h
C
y
A

ALOC



Transformer과 다르게 SSM은 계산의 시간 복잡도가 줄어들지만 행렬 자체는 h(t)와 다르게 고정되기 때문에 계산이 정적이다.

이는 선택적으로 주변을 관찰하는 transformer보다 성능이 떨어질 수 있다.

Mamba는 transformer의 장점과 SSM의 장점을 합쳐서 나온 모델이다.

학습 중 겪은 어려움과 사례 _김동현

ALOC

```
class ImageTransform():
    def __init__(self, resize, mean, std):
        self.data_transform = {
            'train': transforms.Compose([
                transforms.RandomResizedCrop(resize, scale=(0.5, 1.0)),
                transforms.RandomHorizontalFlip(),
                transforms.ToTensor(),
                transforms.Normalize(mean, std)
            ]),
            'val': transforms.Compose([
                transforms.Resize(256),
                transforms.CenterCrop(resize),
                transforms.ToTensor(),
                transforms.Normalize(mean, std)
            ])
        }

    def __call__(self, img, phase):
        return self.data_transform[phase](img)
```

코랩에 책의 코드를 입력하면 자동완성이 되었고
공부 효율이 떨어지는 현상 발생

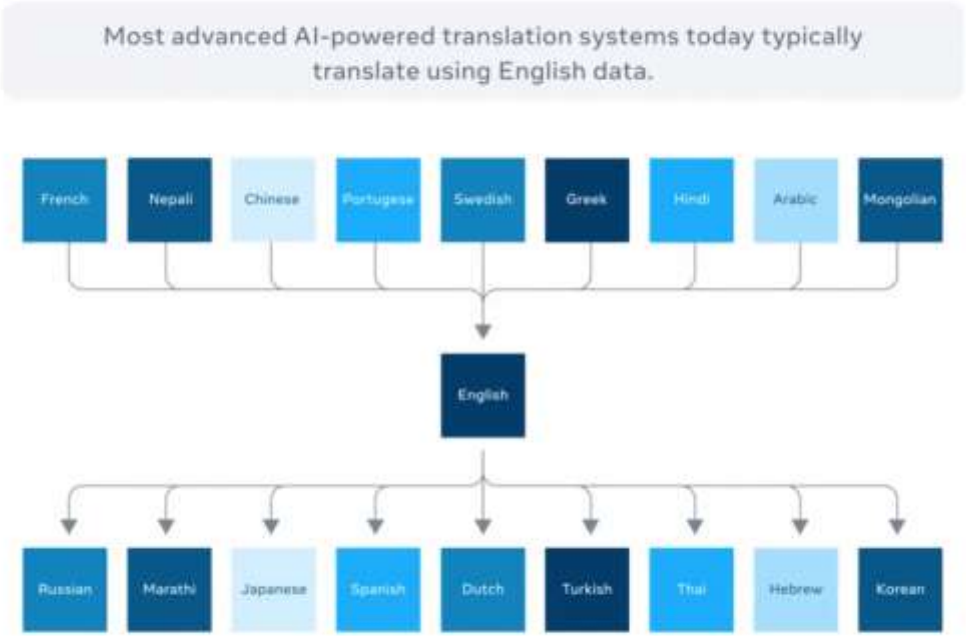
인터넷에서 실전 코드를 가져와서 직접 탐구하는
방법으로 공부 효율을 올림

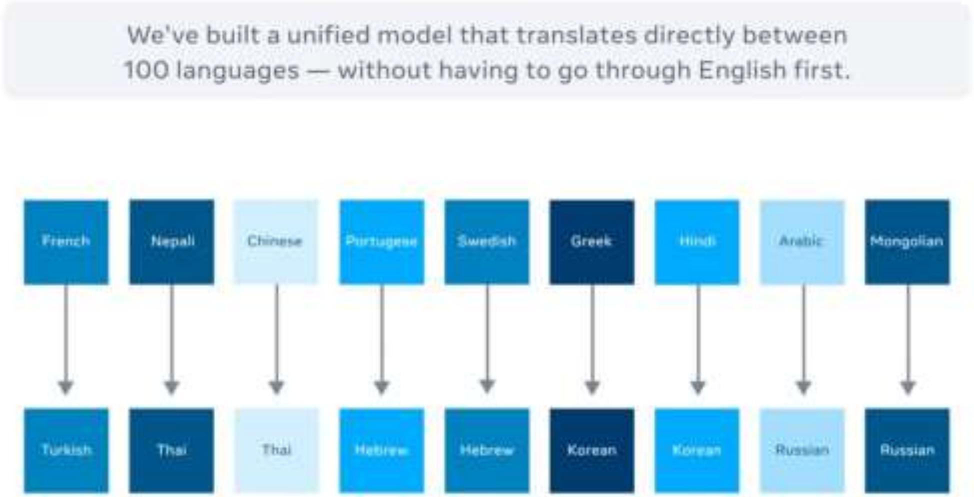
ALOC

활동 중 느낀 점 및 회고 _김동현

깊이 있는 탐구 사례 _박도현

ALOC

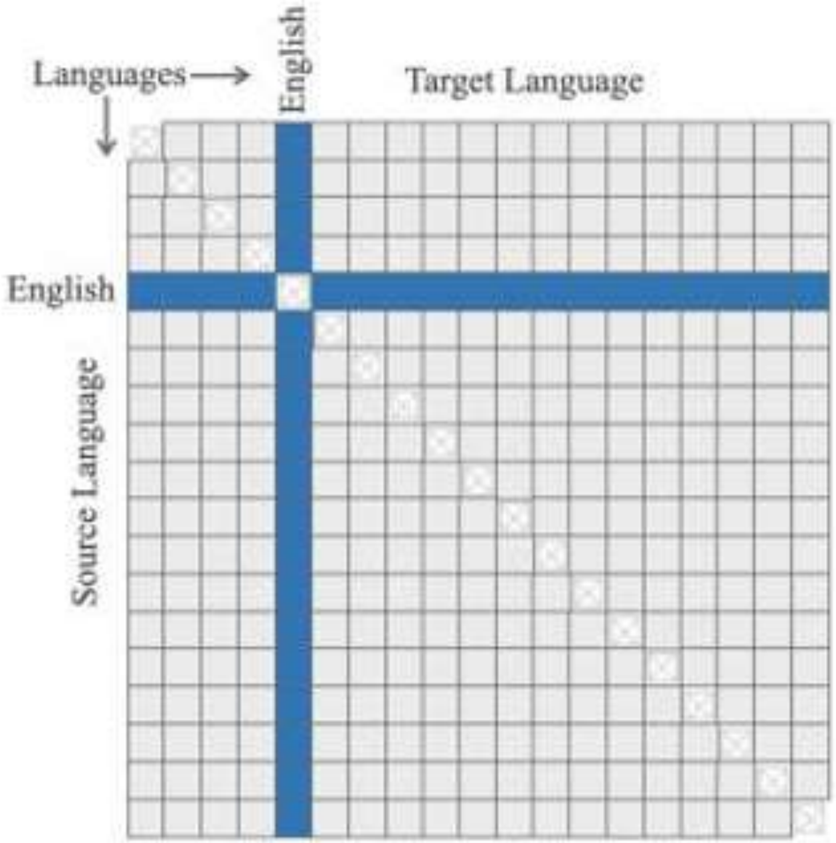




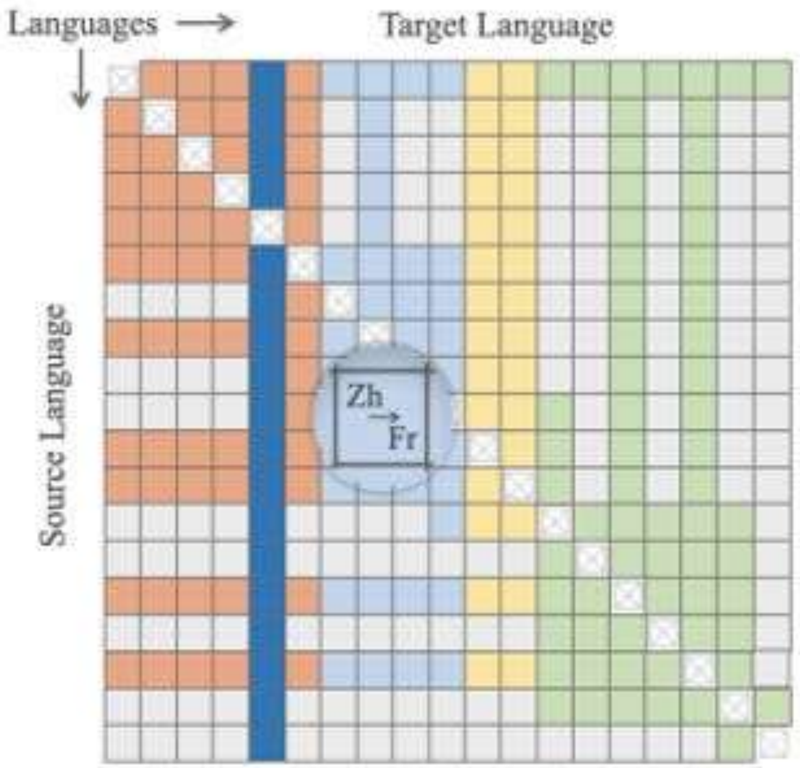
Tranformer encoder-decoder구조를 이용한 번역 특화 모델

ALOC

English-Centric Multilingual

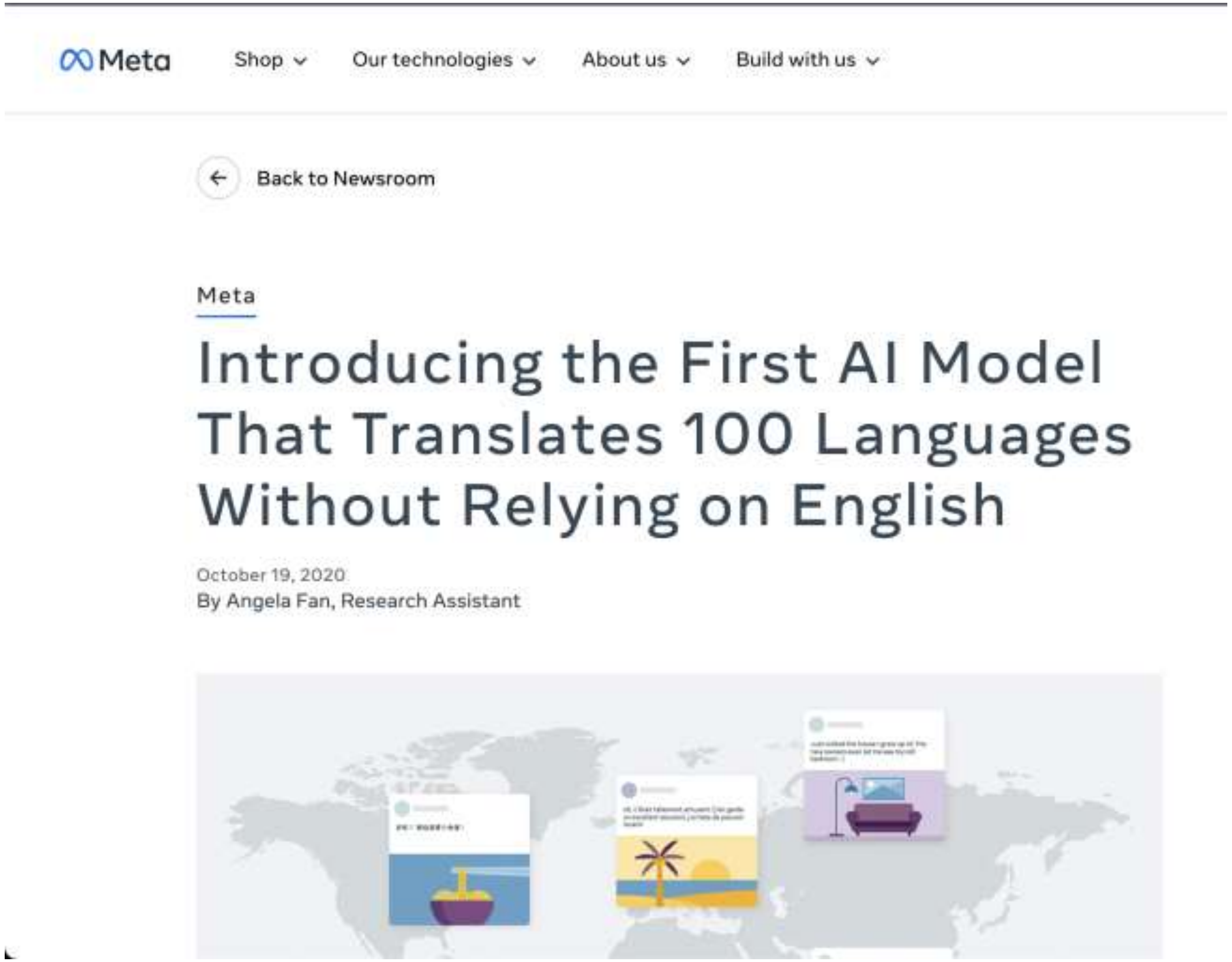


M2M-100: Many-to-Many Multilingual Model



학습 중 겪은 어려움과 사례 _박도현

ALOC



Meta Shop Our technologies About us Build with us

Back to Newsroom

Meta

Introducing the First AI Model That Translates 100 Languages Without Relying on English

October 19, 2020
By Angela Fan, Research Assistant

아직은 자료가 많지 않거나
자료별로 말하는 내용이 다른 점.
너무 빠르게 기술이 변하고 있는 점이
학습하기 어려웠다.

ALOC

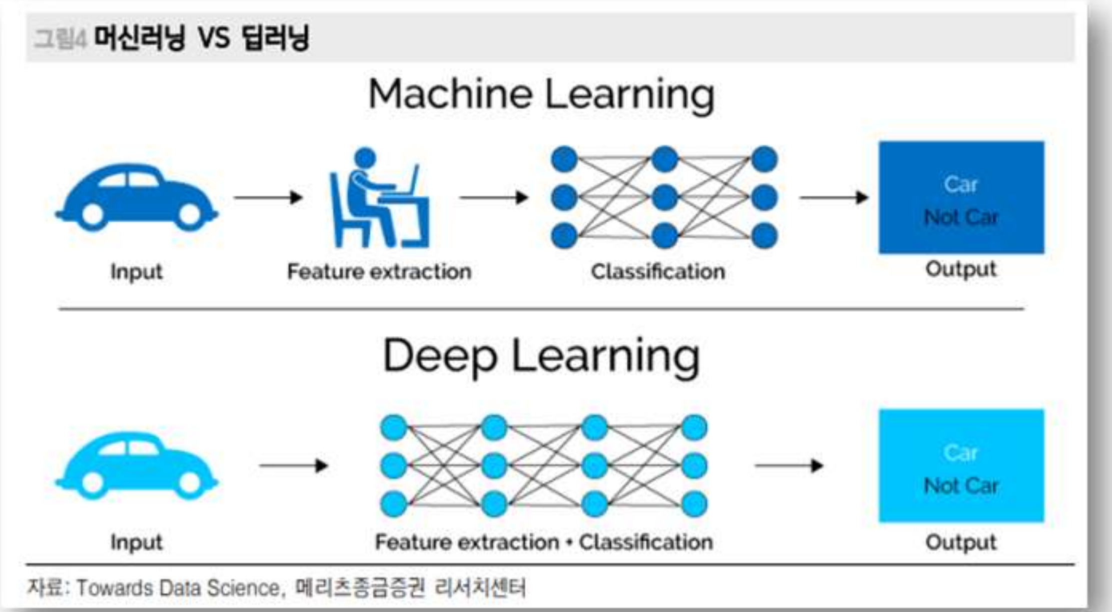
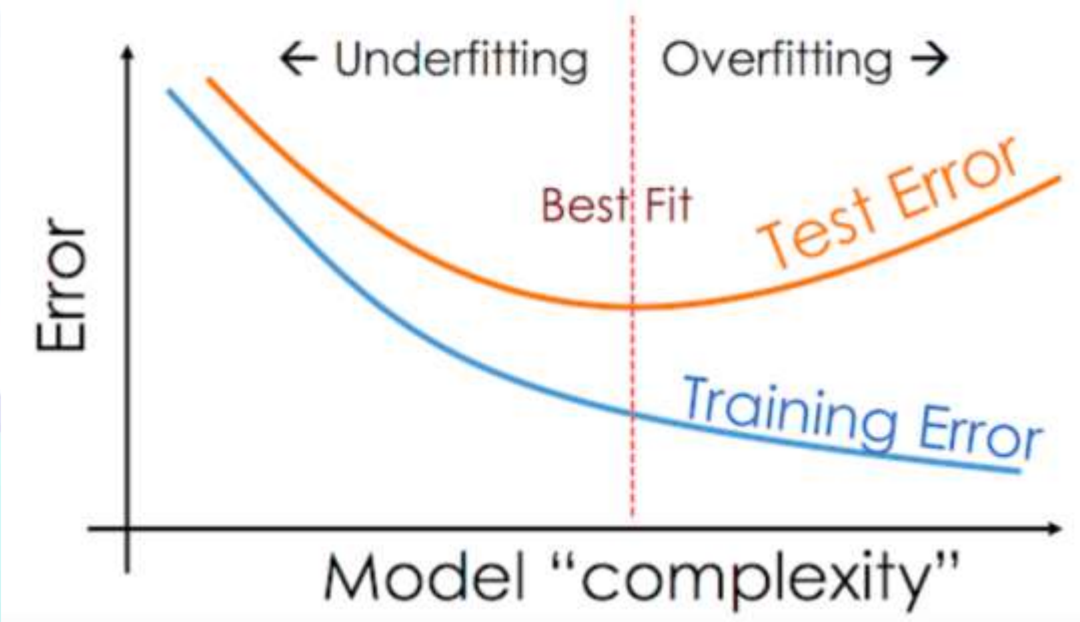


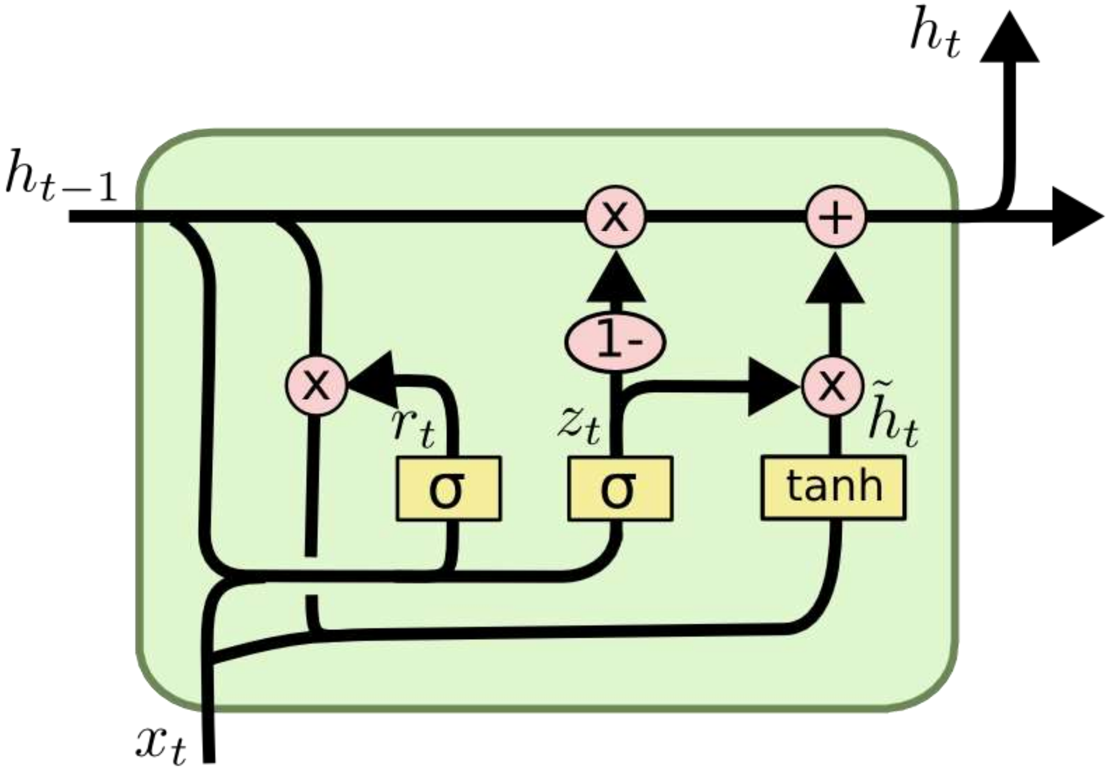
그림4 머신러닝 VS 딥러닝
Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output
자료: Towards Data Science, 메리츠종금증권 리서치센터


← Underfitting
Overfitting →
Best Fit
Test Error
Training Error
Error
Model "complexity"

활동 중 느낀 점 및 회고 _박도현

깊이 있는 탐구 사례 _김준수

RNN / LSTM



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

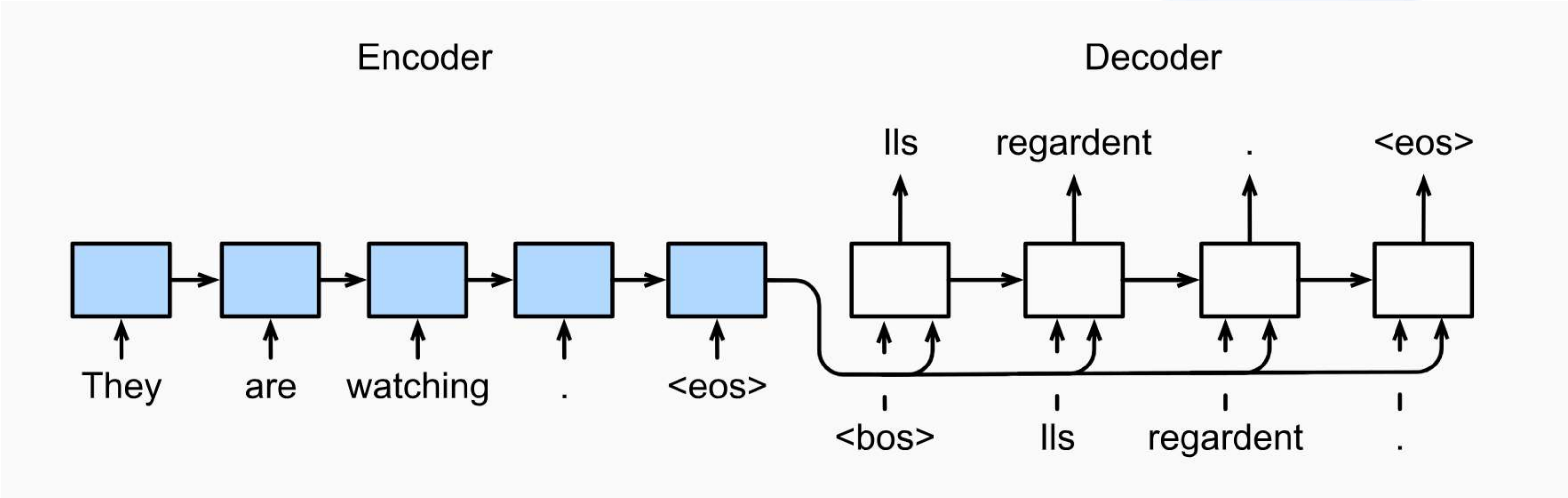
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

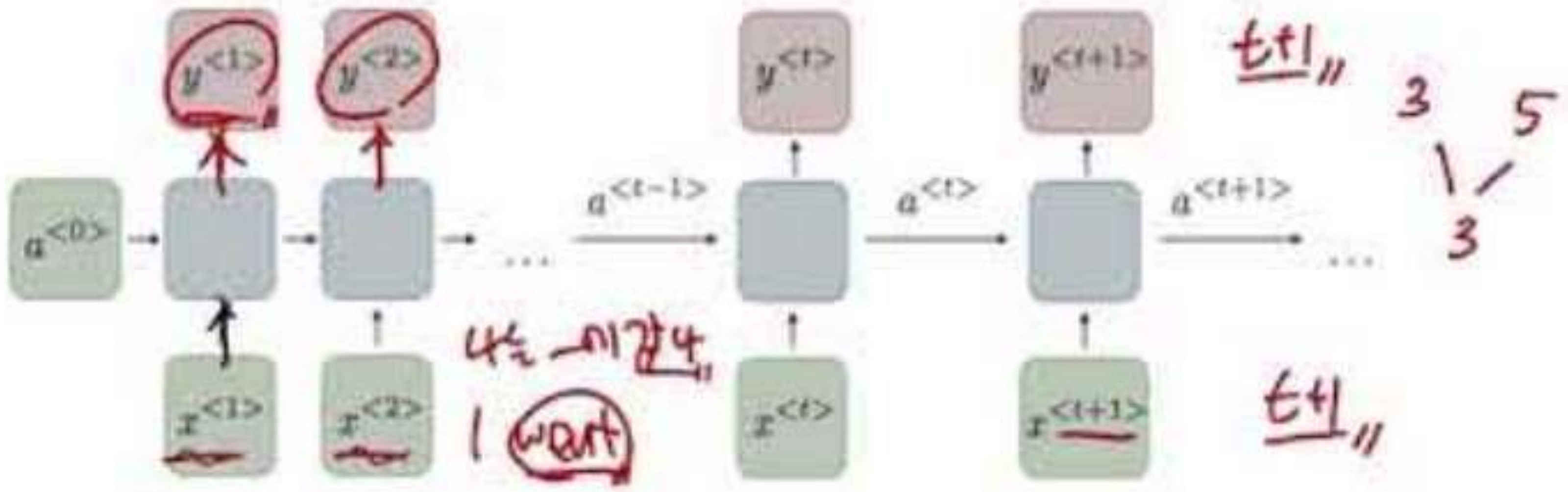
Credit : https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr

Sequence to Sequence Learning with Neural Networks



Credit : https://d2l.ai/chapter_recurrent-modern/seq2seq.html seq2seq presentation

RNN in machine translation



Hypothesize that input size and output size are the same.

Inputs and targets are encoded as vectors of fixed dimensionality.

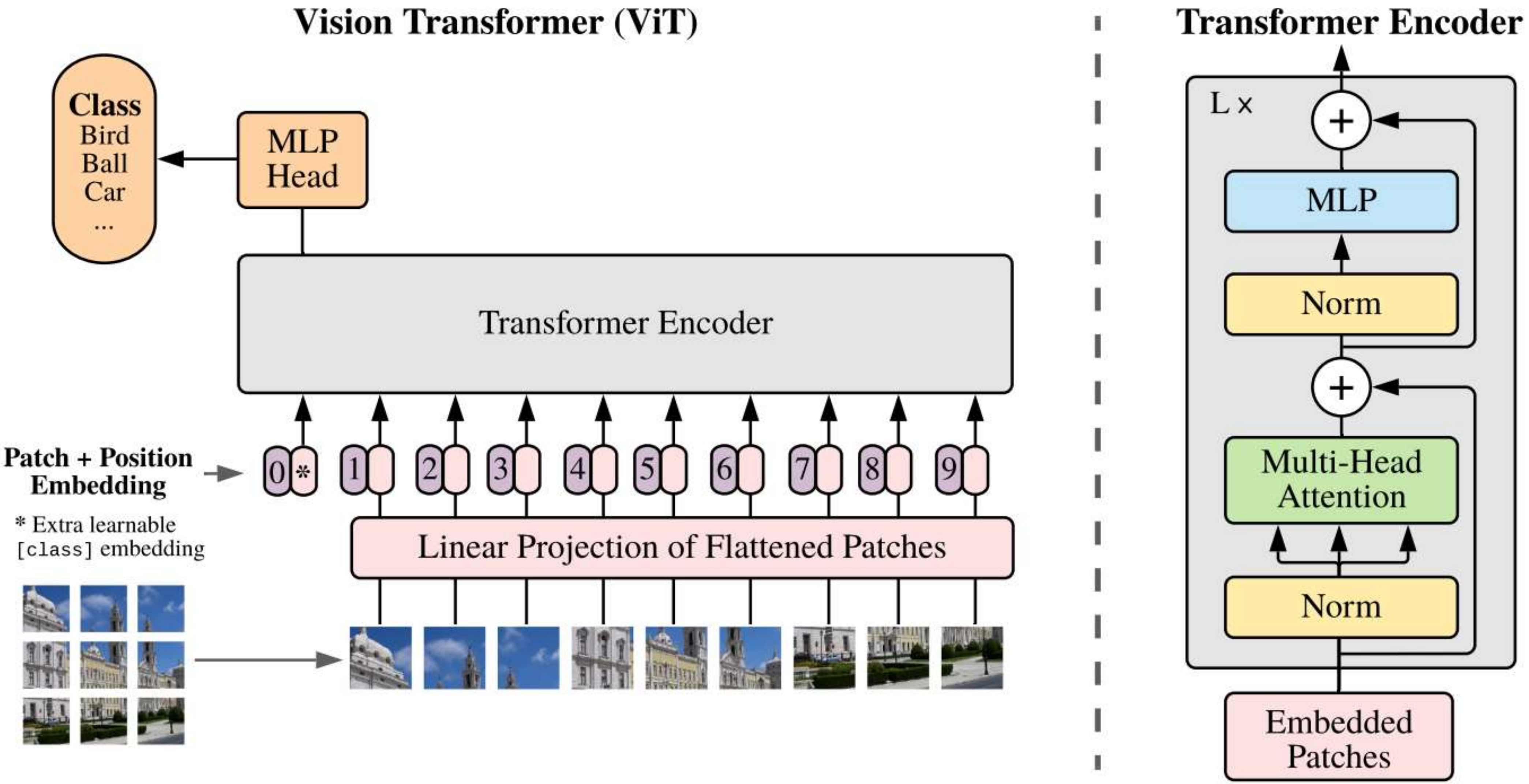
It may not capture the order of sequence.

Credit : https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

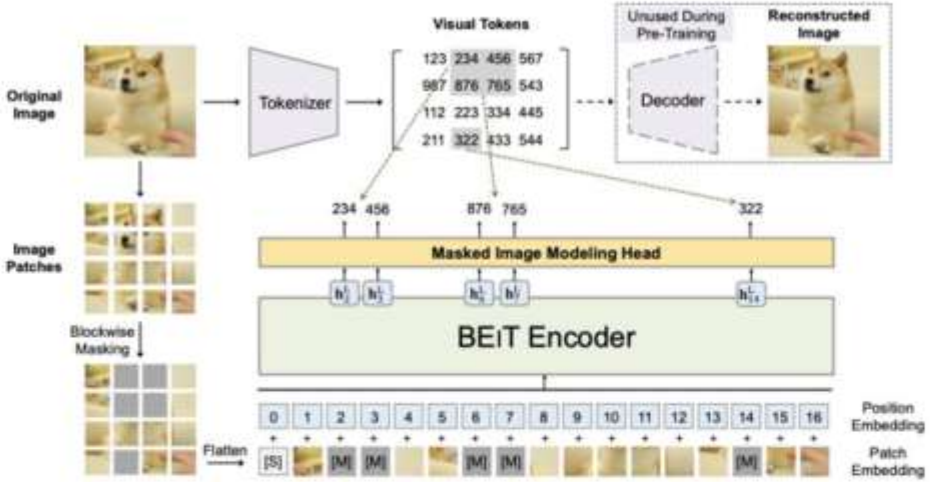
활동 중 느낀 점 및 회고 _김준수

깊이 있는 탐구 사례 _유영호

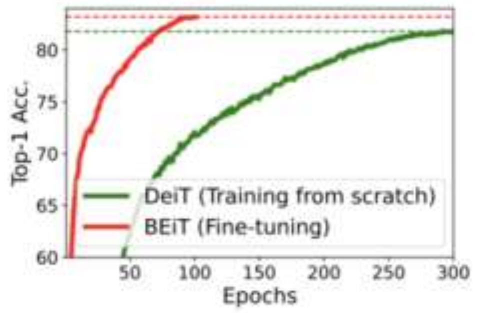
ALOC


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L ×
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

- BEiT: BERT Pre-Training of Image Transformers
  - Before pre-training, learn an "image tokenizer" via VQ-VAE/GAN, where an image is tokenized into discrete visual tokens
    - Similar approaches have been used for image generation, such as DALLE, Parti.
  - Randomly masking image patches, pre-train the model to predict masked visual tokens
  - Can be understood as knowledge distillation between the image tokenizer and the BEiT encoder, but the latter only sees partial of the image

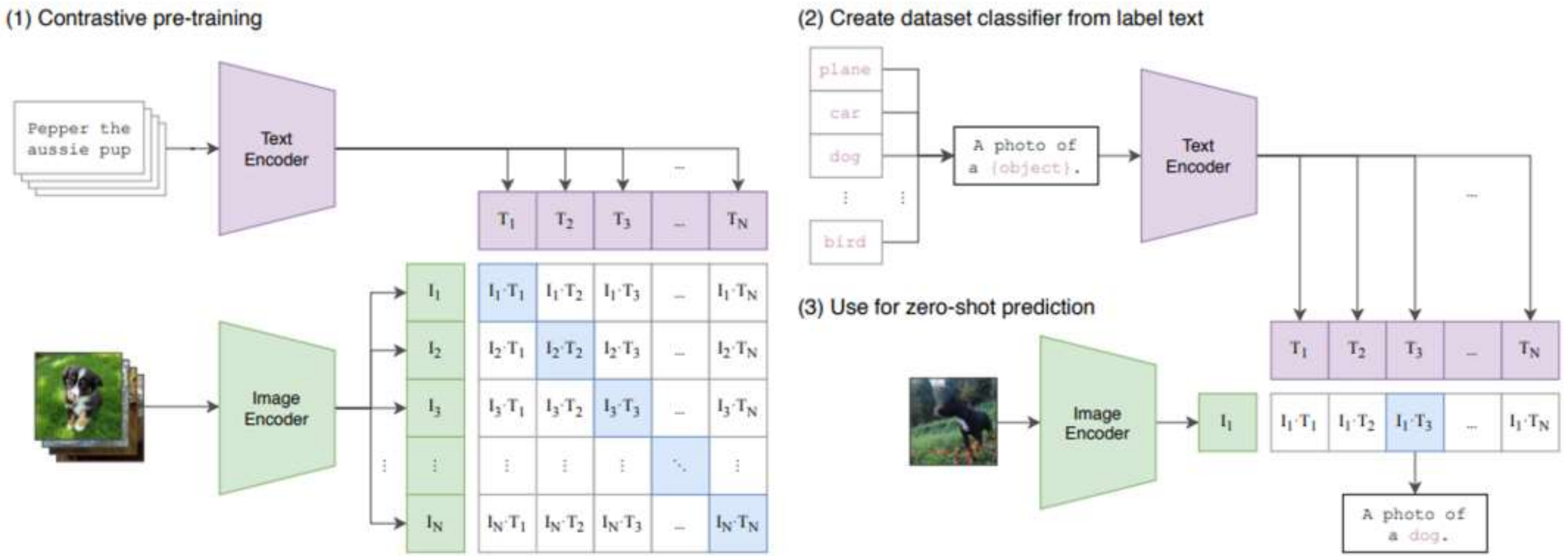


Strong model finetuning performance



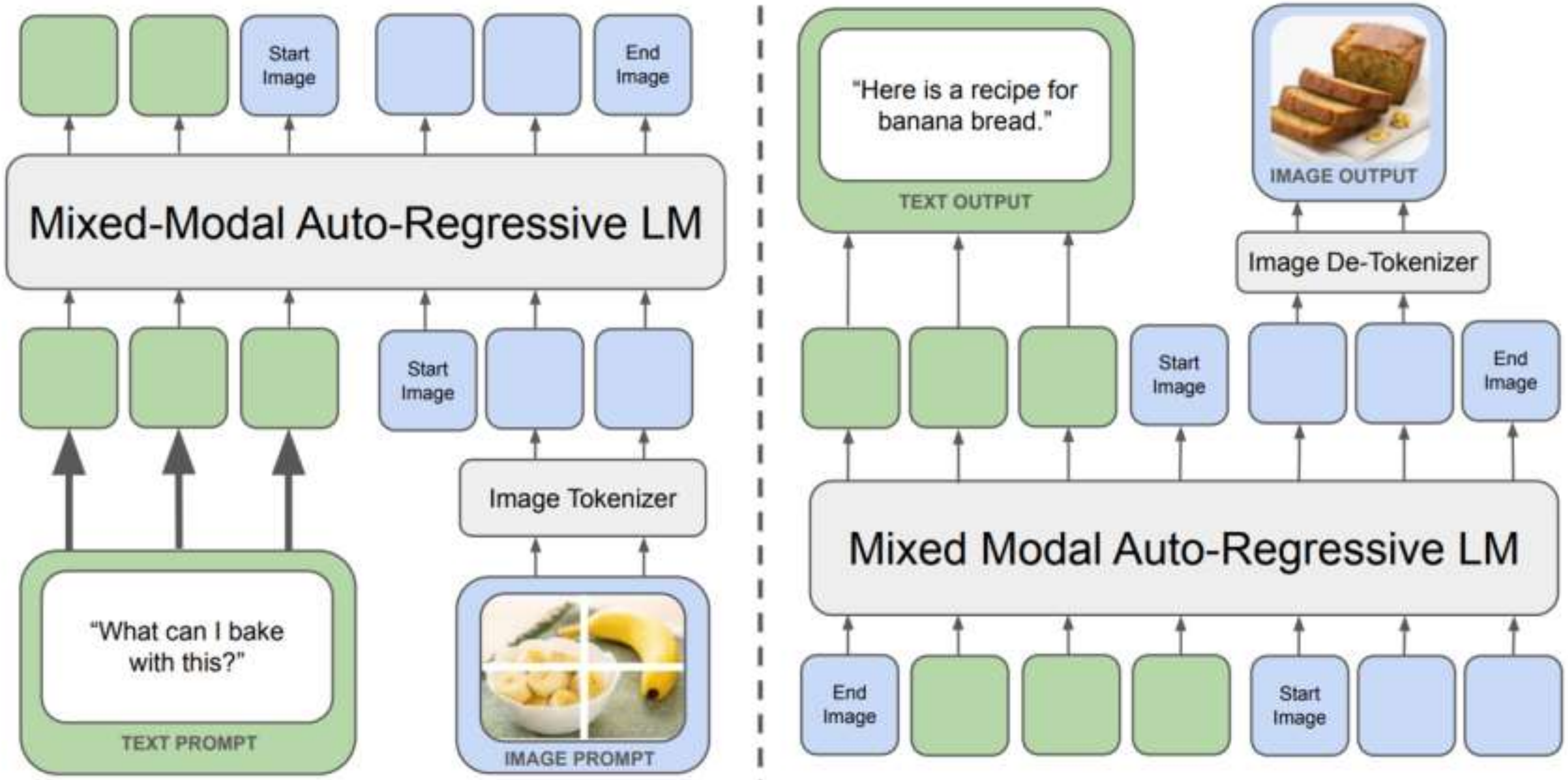
[1] BEiT: BERT Pre-Training of Image Transformers, ICLR 2022
[2] iBOT: Image BERT Pre-Training with Online Tokenizer, ICLR 2022

ALOC



Learning transferable visual models from natural language supervision

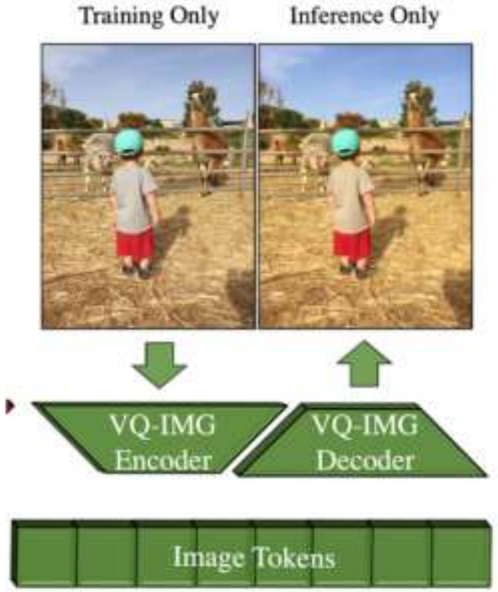
ALOC

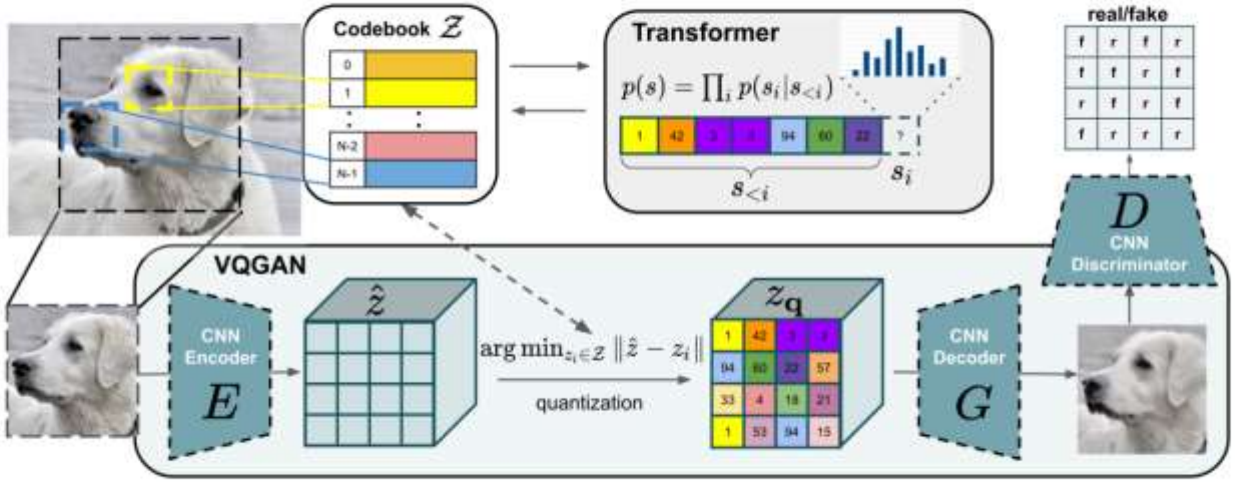


(a) Mixed-Modal Pre-Training (b) Mixed-Modal Generation

Figure 1 Chameleon represents all modalities — images, text, and code, as discrete tokens and uses a uniform transformer-based architecture that is trained from scratch in an end-to-end fashion on ~10T tokens of interleaved mixed-modal data. As a result, Chameleon can both reason over, as well as generate, arbitrary mixed-modal documents. Text tokens are represented in green and image tokens are represented in blue.

ALOC





2.1 Tokenization

Image Tokenization We train a new image tokenizer based on Gafni et al. (2022), which encodes a 512×512 image into 1024 discrete tokens from a codebook of size 8192. For training this tokenizer, we use only licensed images. Given the importance of generating human faces, we up-sample the percentage of images with faces during pre-training by 2 times. A core weakness of our tokenizer is in reconstructing images with a large amount of text, therefore upper bounding the capability of our models, when it comes to heavy OCR-related tasks.

ALOC

Prompt: I'm bored. Could you show me some cool, quirky-looking birds? Also tell me about them in 2–3 sentences.

Response: Absolutely! Here are three birds with some unique features that you might find interesting:

Keel-Billed Toucan The Keel-Billed Toucan is a brightly colored bird native to South American rainforests and woodlands. Its most distinctive feature is its colorful beak, which can grow up to seven inches long and has a distinctive, curved shape. The beak is used for plucking fruit from trees and for social displays. Despite its large size, the Keel-Billed Toucan is a skilled flyer and can often be seen flitting through the treetops. Here is an image of this vibrant bird: <img>

Puffin Puffins are small, aquatic birds that are found in the northern hemisphere. They have distinctive black and white feathers, as well as a brightly colored beak that is used for attracting mates and defending their territory. Puffins are known for their distinctive "puffin" call, which sounds like a loud, guttural chuckle. They feed on fish, which they catch by diving into the water from the air. Puffins are colonial birds, and they breed in large colonies on rocky cliffs and islands. Here is an image of this adorable bird: <img>

Golden Pheasant The Golden Pheasant is a brightly colored bird native to the forests of China. It is a member of the pheasant family and is known for its striking plumage, which features a golden-yellow body, red face, and green tail feathers. The Golden Pheasant is a ground-dwelling bird, and it feeds on a diet of insects, seeds, and fruits. It is a popular game bird, and is often hunted for its meat and feathers. In captivity, the Golden Pheasant is sometimes kept as an ornamental bird, due to its striking appearance. Here is an image of this beautiful bird: <img>

I hope these images and descriptions have helped to brighten your day! If you have any other questions or requests, please don't hesitate to ask.

Prompt: What can I bake with this? Give me the full recipe and show me an image with the final dish. <img>

Response: Bananas are a versatile fruit that can be used in various baked goods, such as muffins, bread, cakes, and cookies. They add natural sweetness, moisture, and a subtle banana flavor to the finished product. Additionally, bananas are rich in potassium, fiber, and other nutrients, making them a healthy addition to your baking repertoire.

Here is a simple recipe for banana bread that uses 3 ripe bananas:

Ingredients:
- 3 ripe bananas, mashed
- 1 1/2 cups all-purpose flour
- 1 tsp baking powder
- 1/2 tsp baking soda
- 1/2 tsp salt
- 1/4 cup unsalted butter, softened
- 1 cup granulated sugar
- 2 large eggs
- 1 tsp vanilla extract
- 1/2 cup chopped walnuts (optional)

Instructions:
- Preheat your oven to 350°F (180°C). Grease a 9x5-inch loaf pan and set it aside.
- In a medium bowl, whisk together the flour, baking powder, baking soda, and salt. Set aside.
- In a large mixing bowl, use an electric mixer to cream together the butter and sugar until light and fluffy, about 2-3 minutes.
- Beat in the eggs one at a time, followed by the mashed bananas and vanilla extract.

- Gradually mix in the dry ingredients until just combined, being careful not to overmix.
- If using walnuts, fold them into the batter.
- Pour the batter into the prepared loaf pan and smooth the top.
- Bake for 50-60 minutes, or until a toothpick inserted into the center of the bread comes out clean.
- Remove the bread from the oven and let it cool in the pan for 10 minutes before transferring it to a wire rack to cool completely. <img>

Chameleon: Mixed-Modal Early-Fusion Foundation Models

활동 중 느낀 점 및 회고 _유영호

ALOC

Q&A

ALOC

감사합니다

ALOC