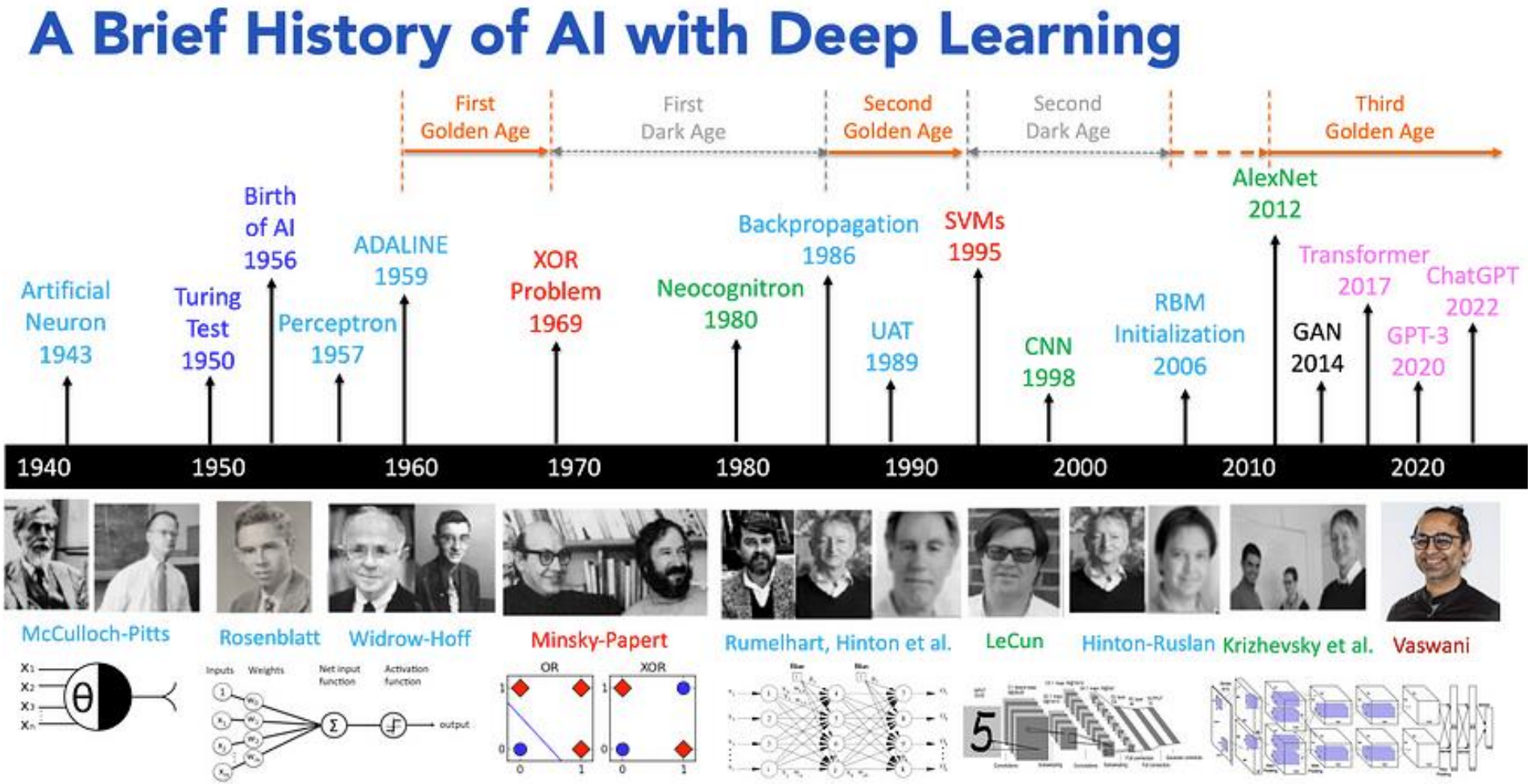
Intro
Transformer

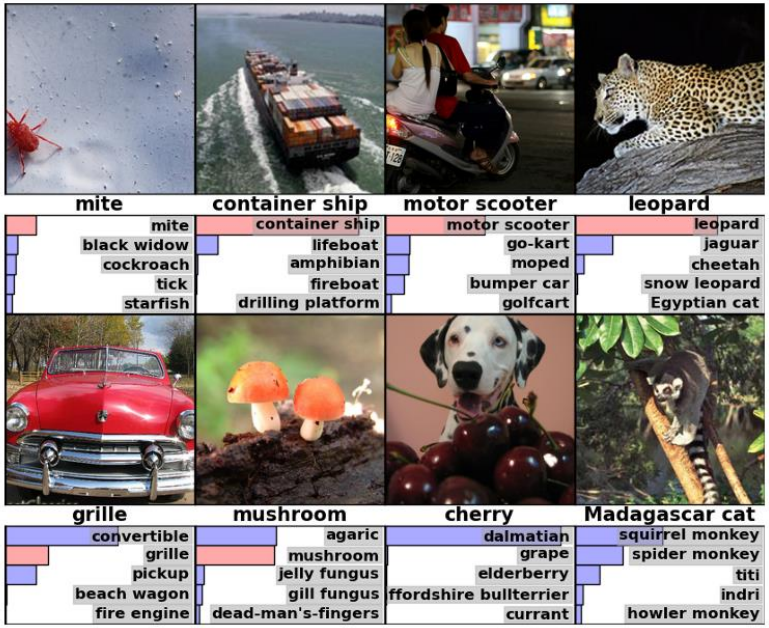
AI History



[A Brief History of AI with Deep Learning | by LM Po | Medium]

AlexNet

Convolutional neural networks running on GPUs (2012) Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, Advances in Neural Information Processing Systems 2012



Deep CNN, ReLU, Dropout

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

ImageNet Classification with Deep Convolutional Neural Networks

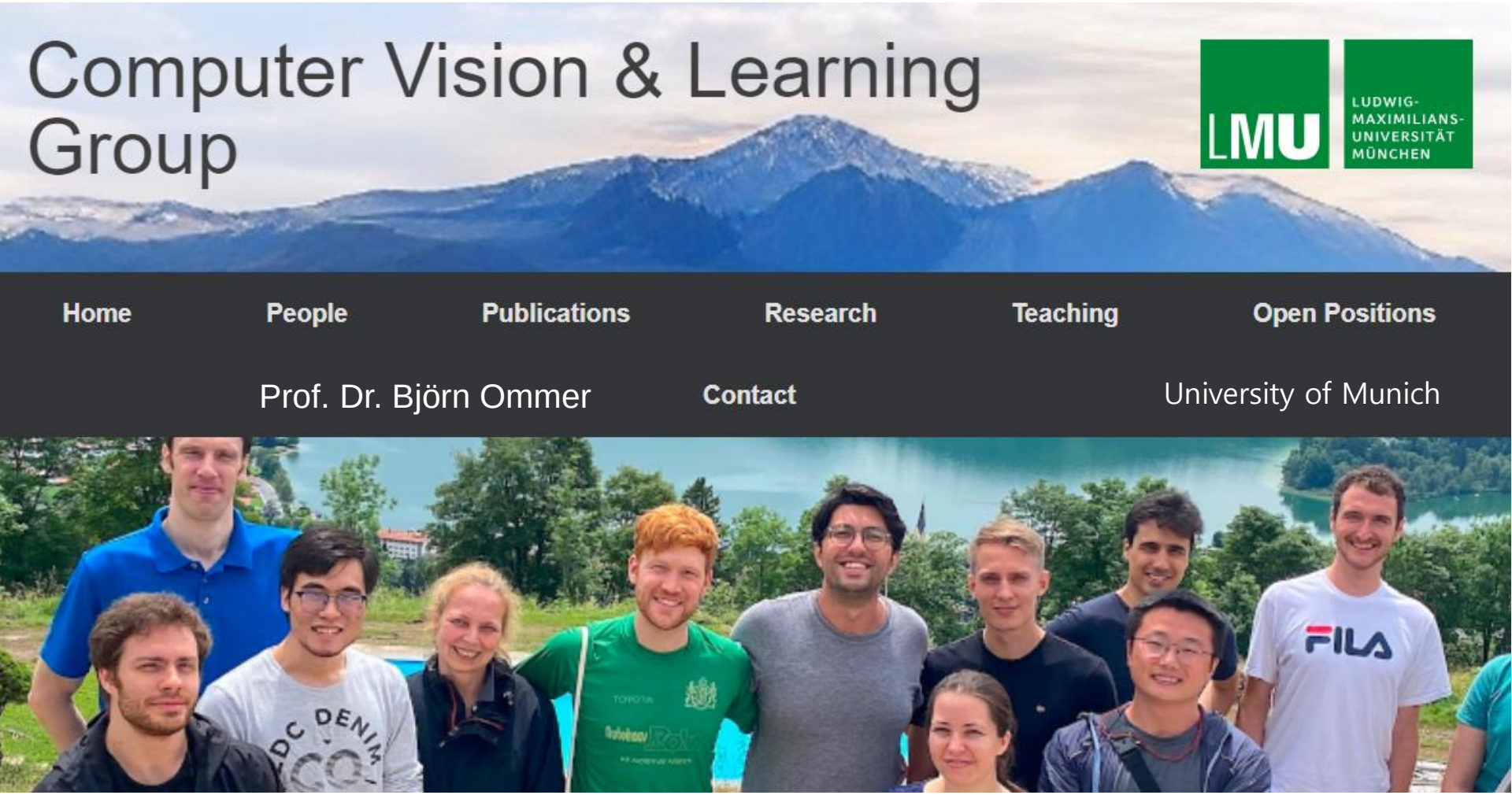
Google images

2013. Google acquired DNN

Facebook AI Research

Yann LeCun.
Chief AI Scientist for Facebook AI Research (FAIR)

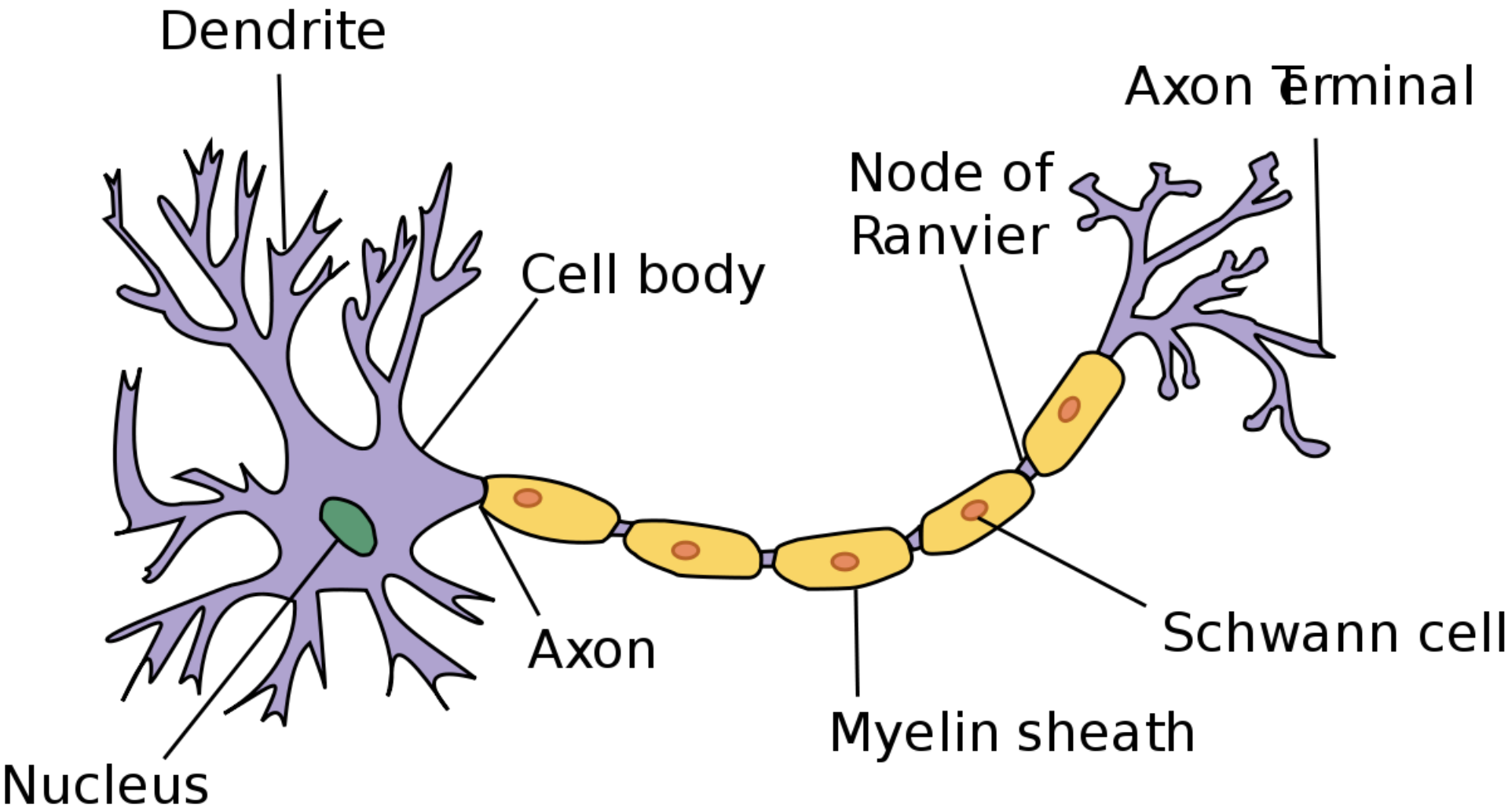
AI divergence



Stable Diffusion Developer [Computer Vision & Learning Group (ommer-lab.com)]

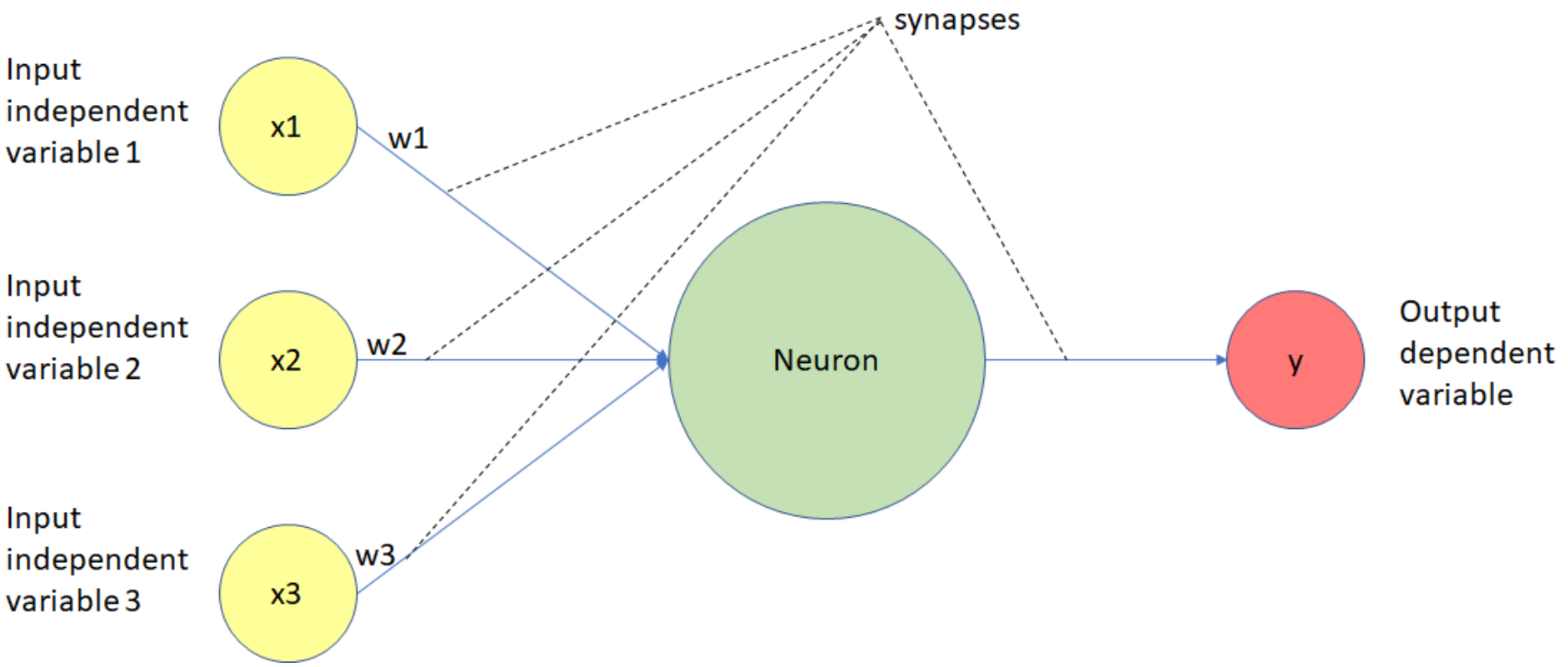
AI
Basic Principle

Deep learning



Deep Learning Neural Networks Explained in Plain English

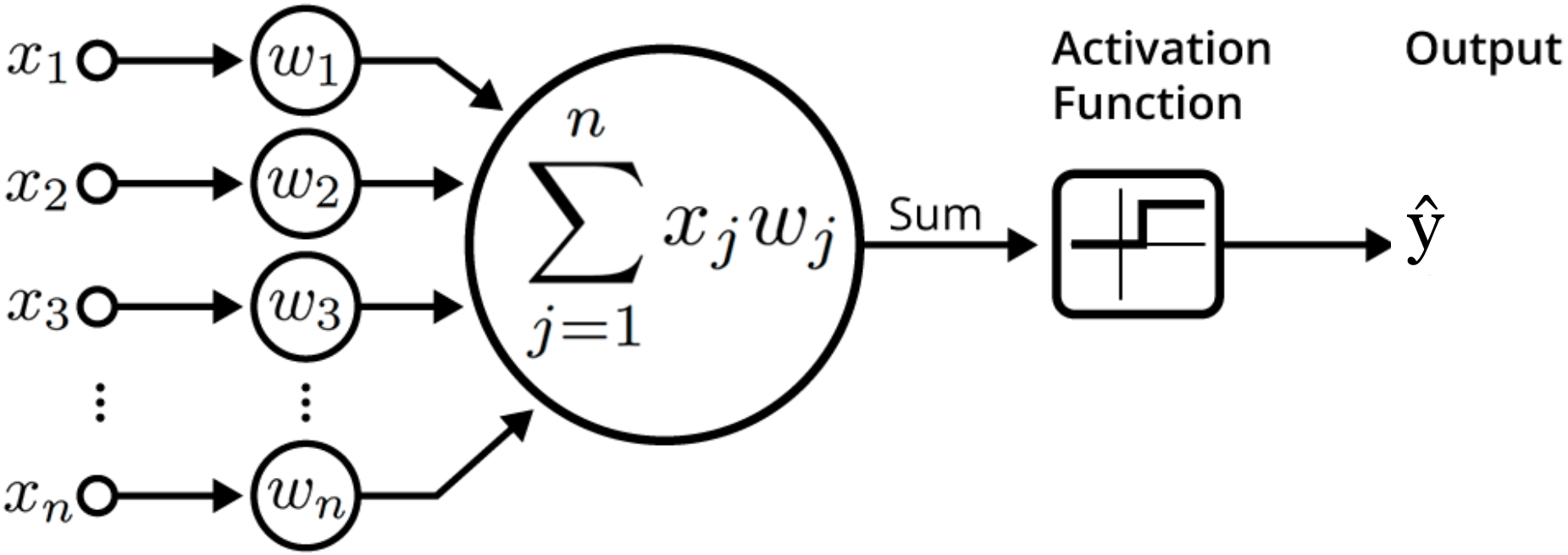
Deep learning



Deep Learning Neural Networks Explained in Plain English

Deep learning

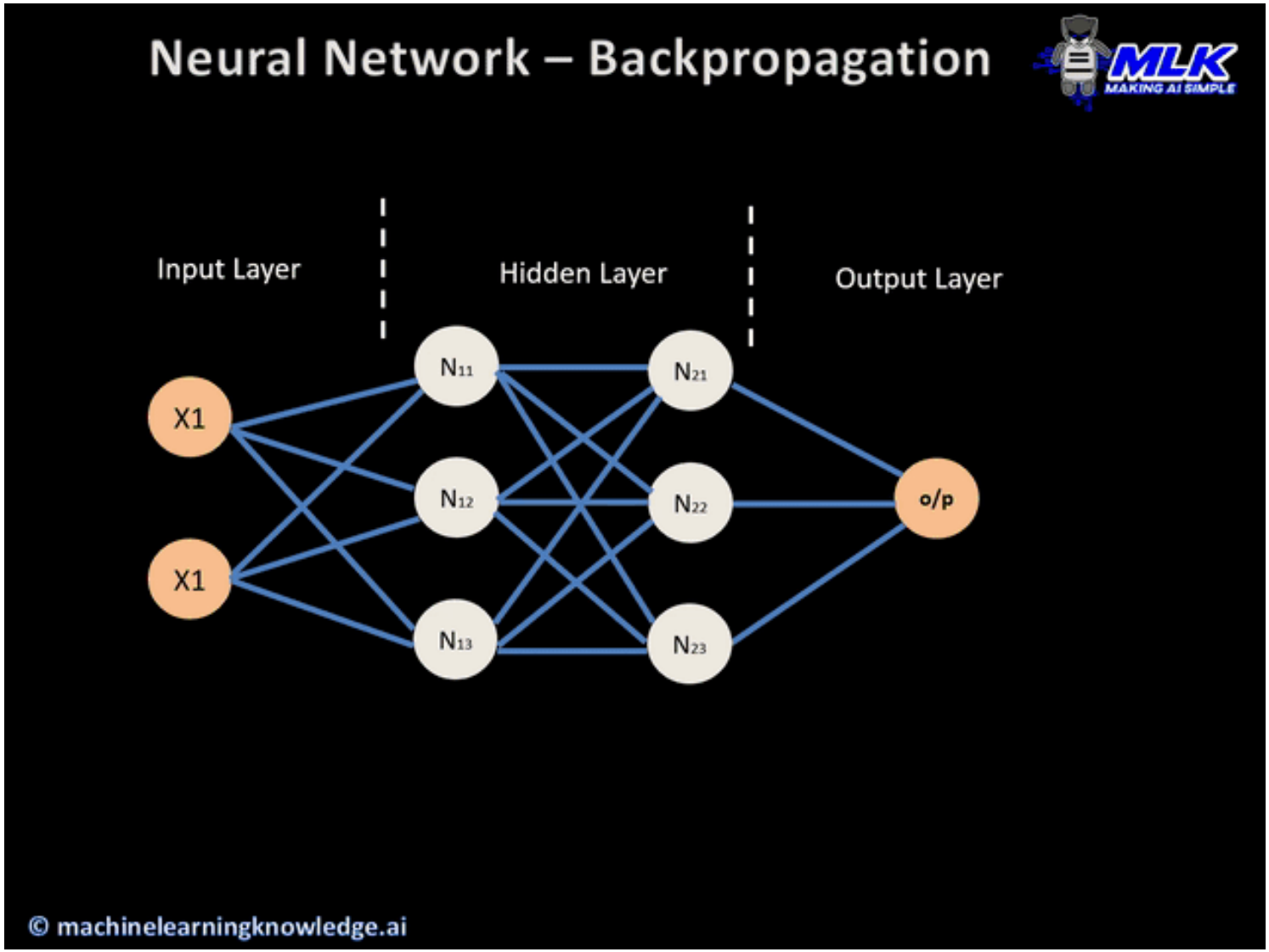
$$\hat{y} = f(W \cdot x + b)$$



$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Deep Learning Neural Networks Explained in Plain English and Becoming Human

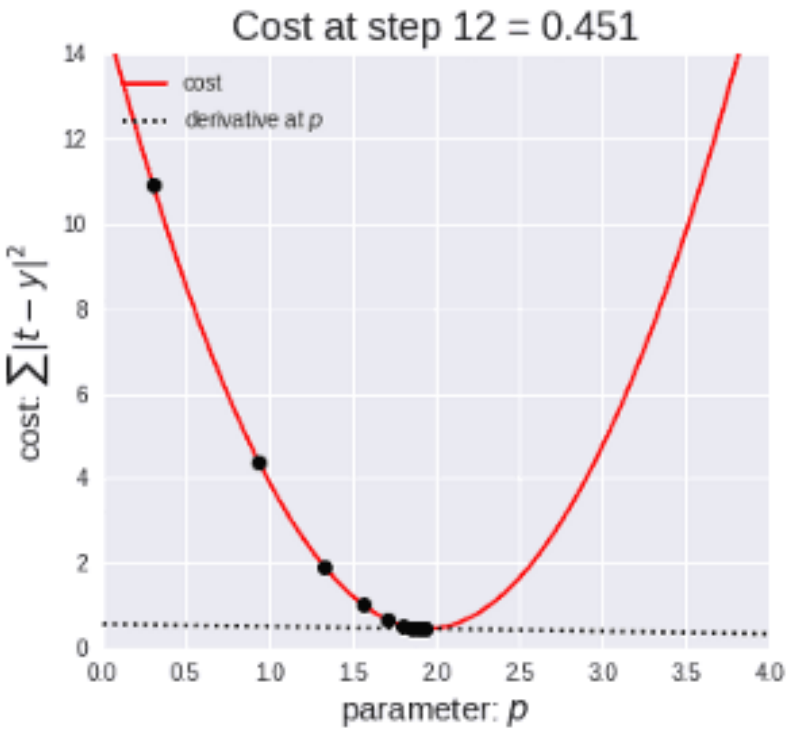
Deep learning

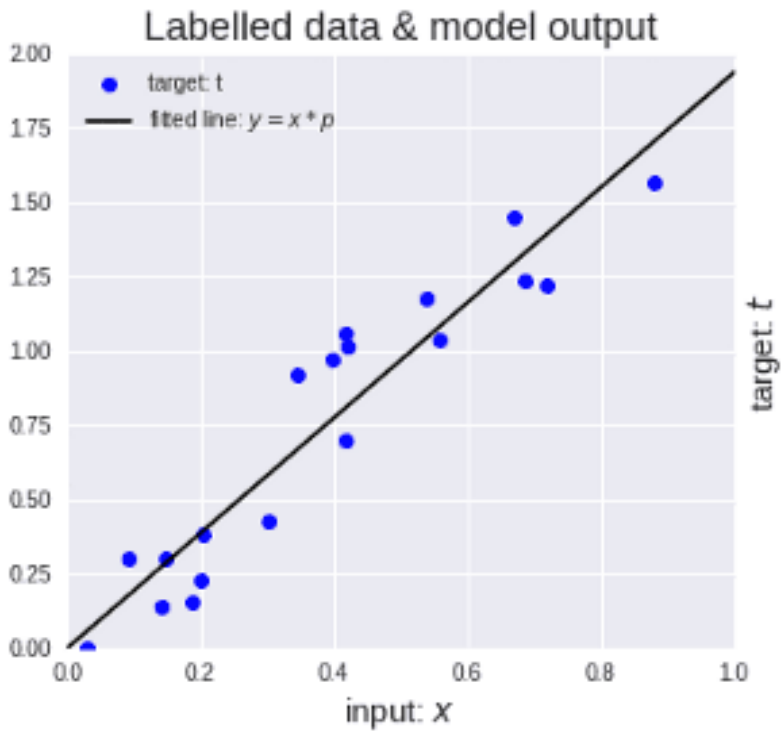


Backpropagation for Dummies. All the math behind, simplest than... | by Diletta Goglia | Analytics Vidhya | Medium

Deep learning

$\hat{y}=f(W\cdot x+b)$ $\quad$ $target=\min_{LOSS}(y-\hat{y})$ $\quad$ $w_{new} = w_{old} + \eta\Delta w$



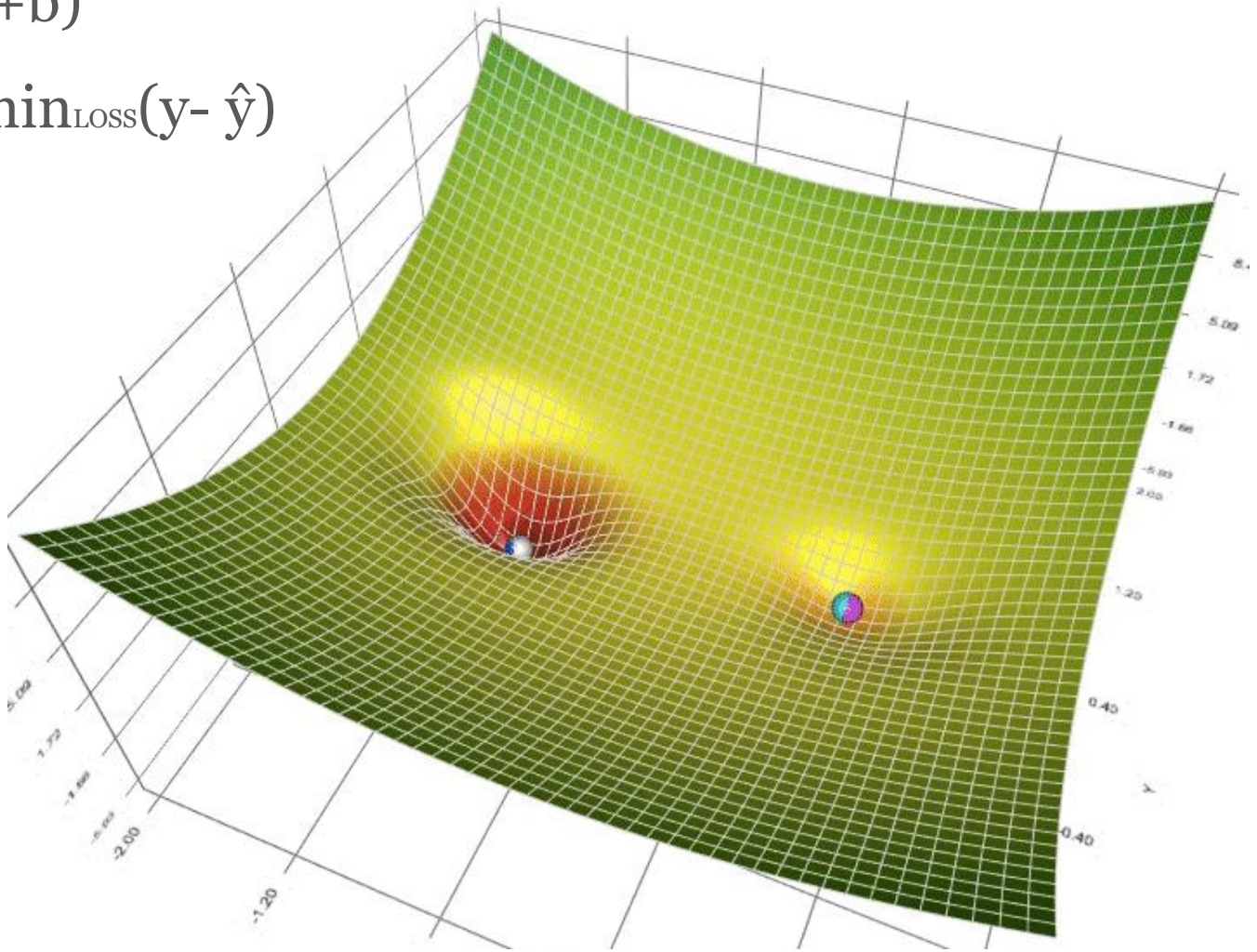


Gradient Descent and Cost Function in Python - Startertutorials

Deep learning

$\hat{y}=f(W\cdot x+b)$

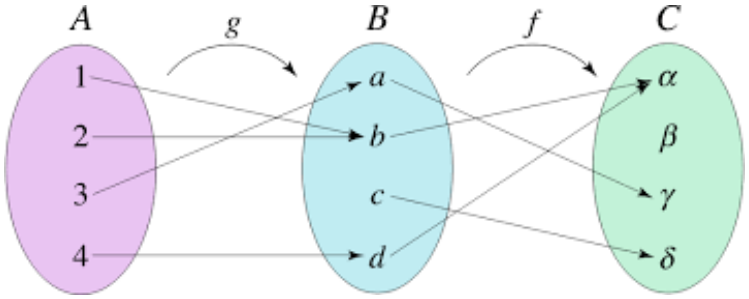
$target=\min_{Loss}(y-\hat{y})$

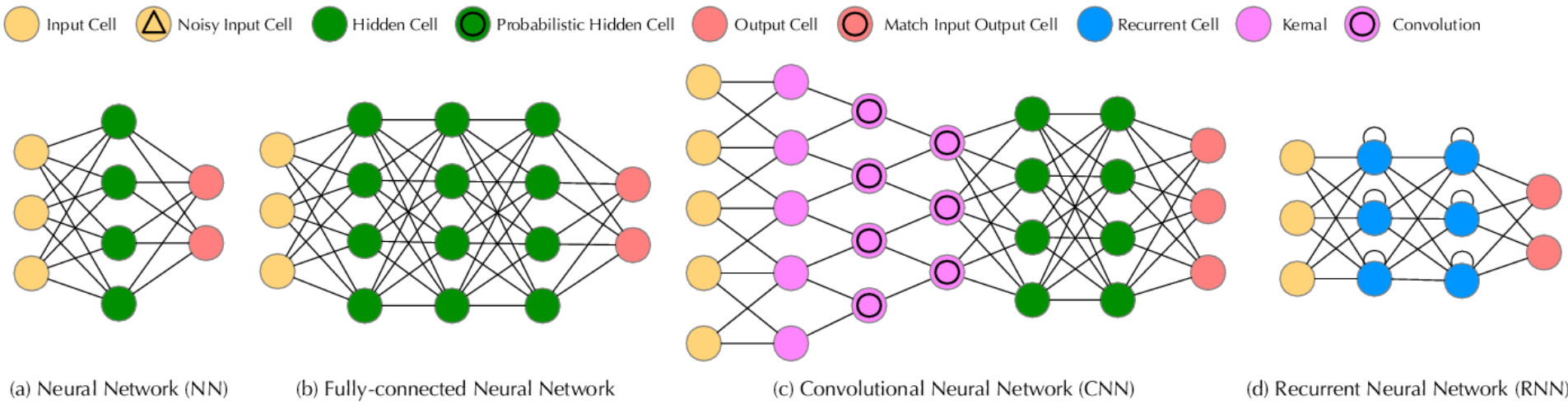


Enhancing Multi-Layer Perceptron Performance: Demystifying Optimizers | by Anand Raj | Towards AI

AI deep learning model network

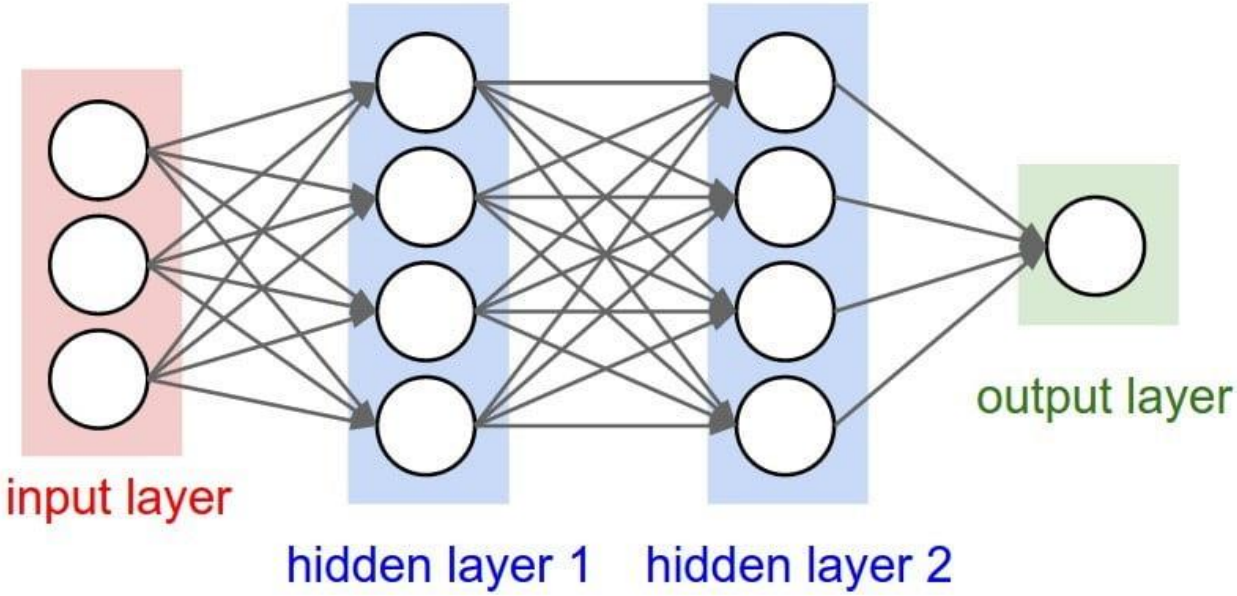
Model = {target loss fun, differential gradient descent func, input feature, output feature}





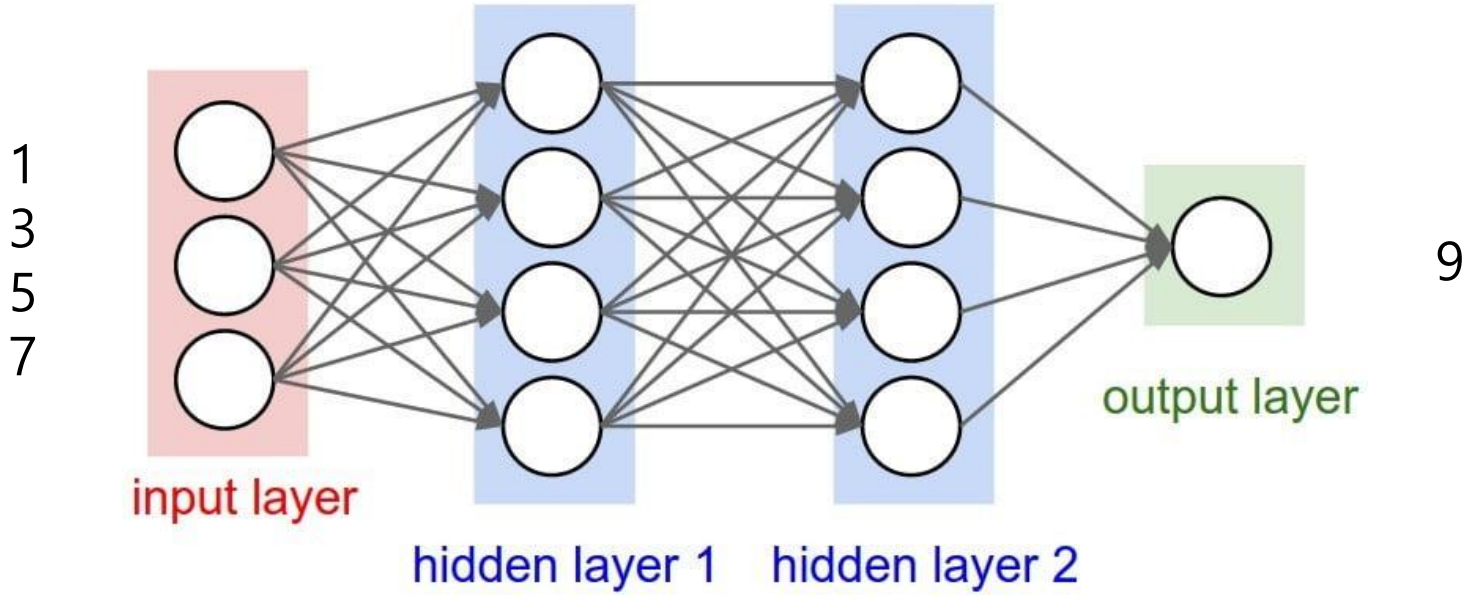
Saravanan Thirumuruganathan, N. Tang, M. Ouzzani, 2018, Data Curation with Deep Learning [Vision]: Towards Self Driving Data Curation, ArXiv

AI & Deep Learning

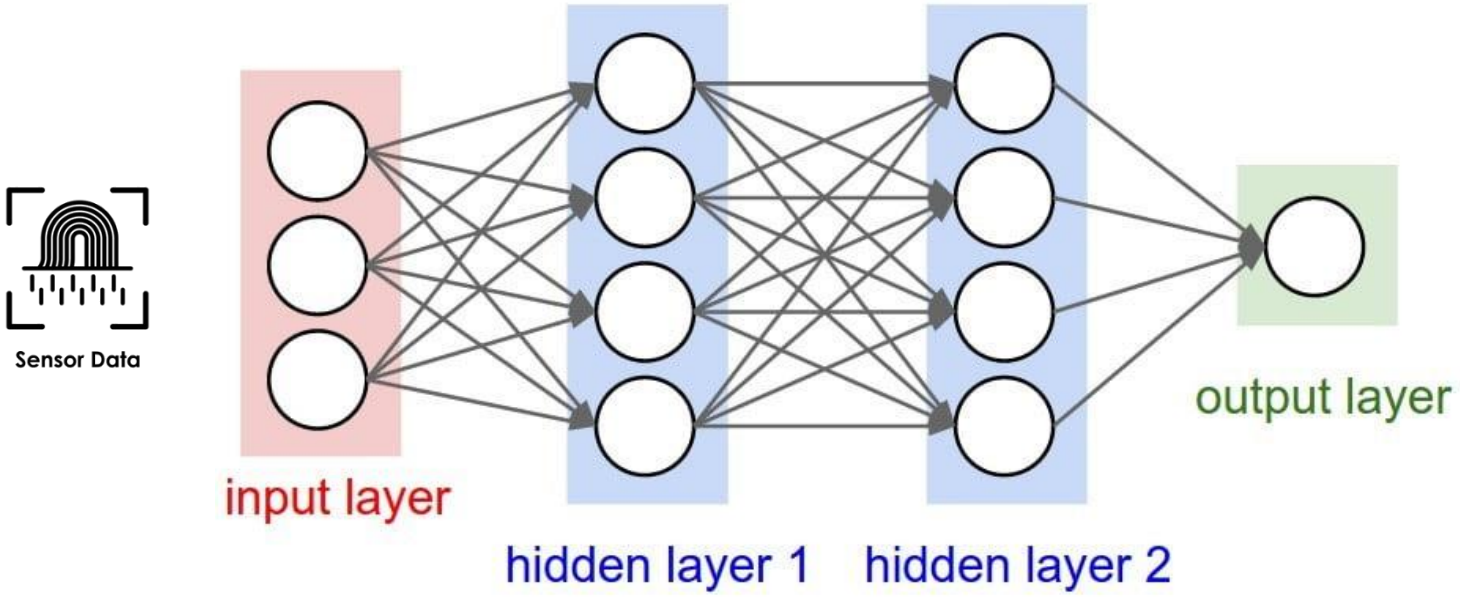


$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

AI & Deep Learning digital numbers



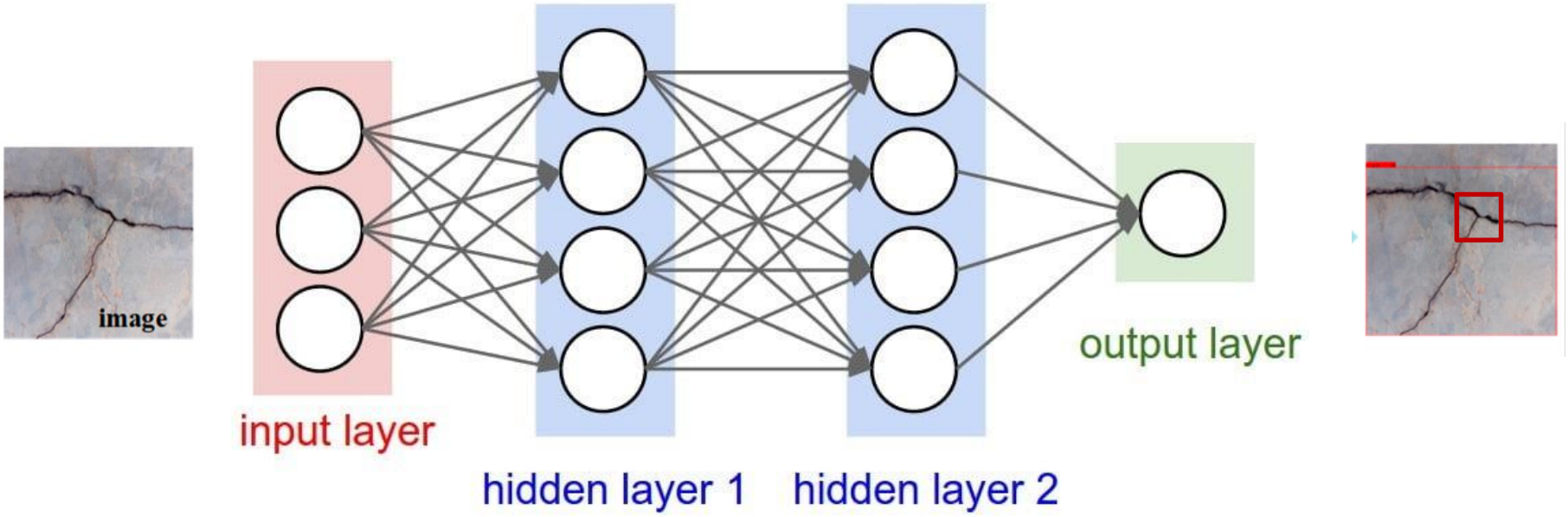
AI & Deep Learning vector



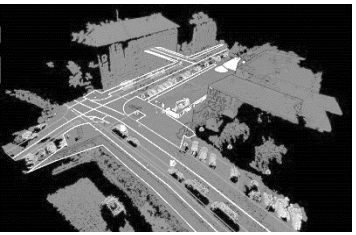
Pattern A
Pattern B
…

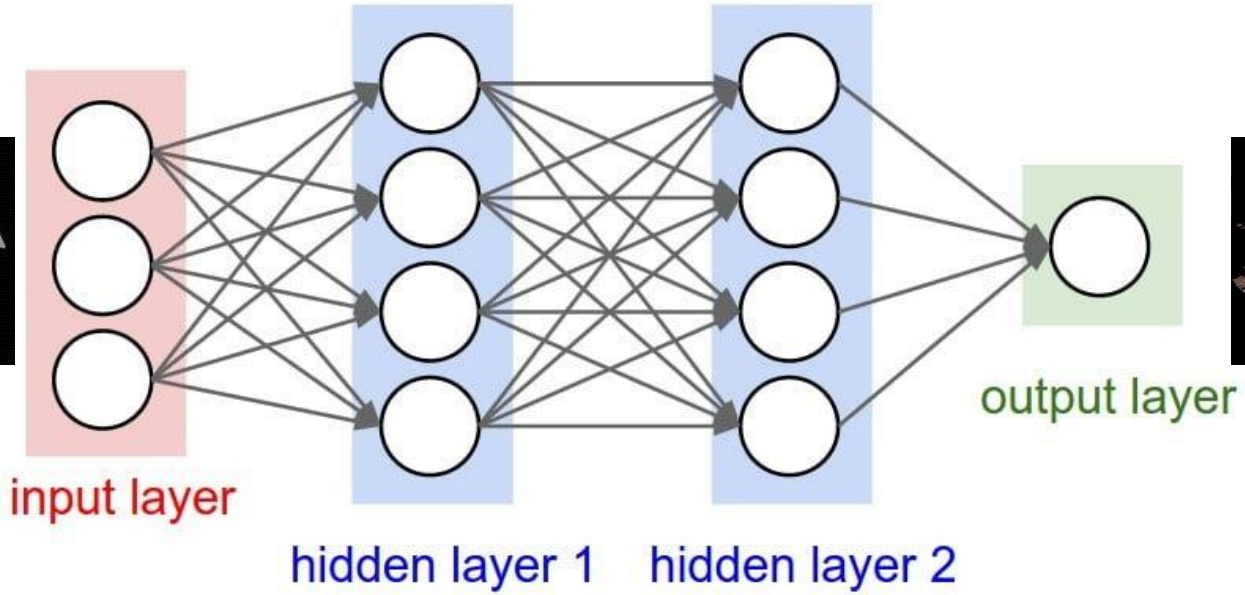
Anomaly

AI & Deep Learning image = 2D matrix

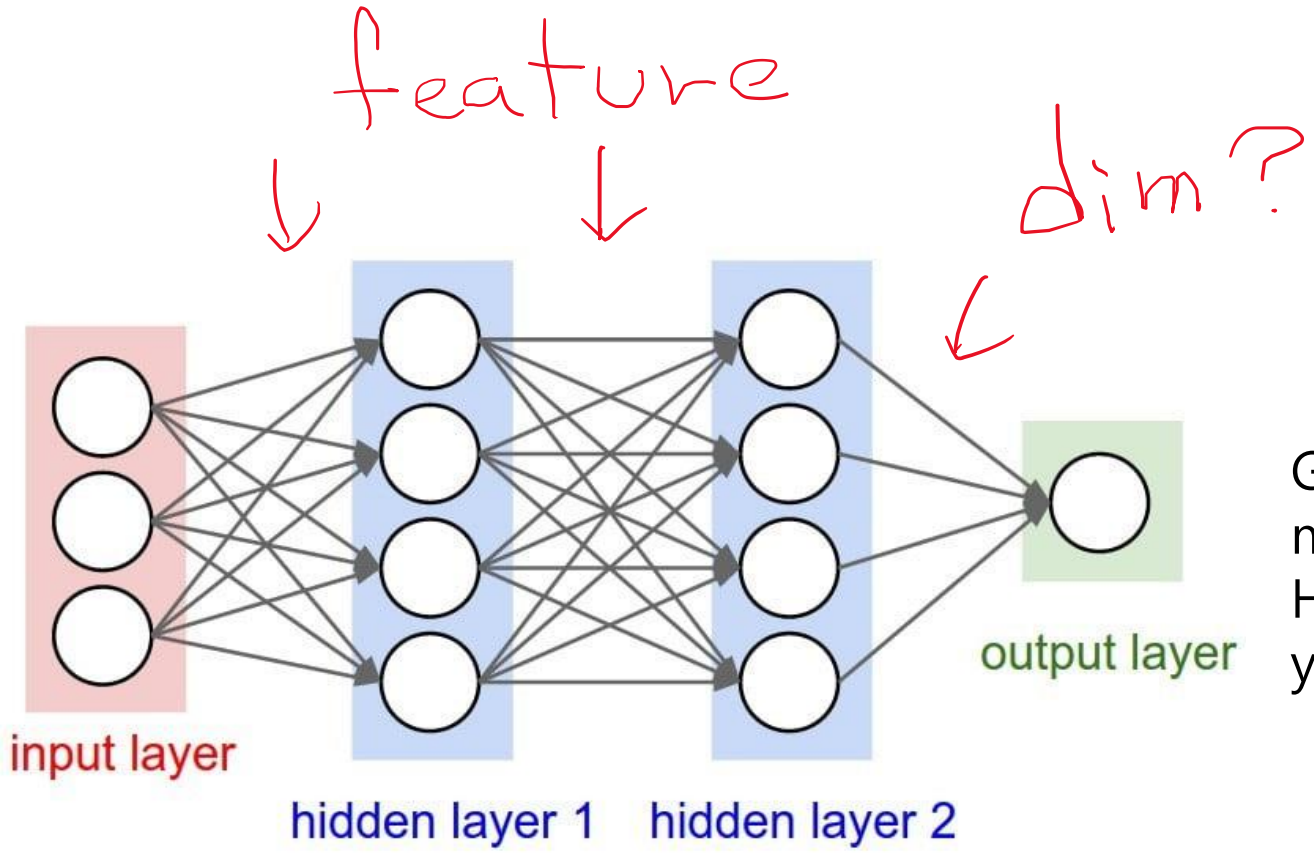


AI & Deep Learning image = n-d matrix





AI & Deep Learning token sequence



Good morning

Good morning! How are you today?

Transfomer

사전, 토큰, 임베딩

딥러닝 모델은 텍스트를 바로 이해하지 못함

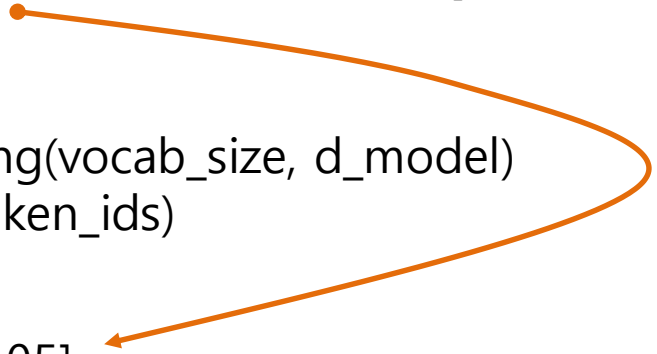
문장 → 토크나이징(tokenizing) → 토큰 ID → 임베딩 벡터

"I am a student" → ["I", "am", "a", "student"]

["I", "am", "a", "student"] → [101, 2023, 1037, 3076]

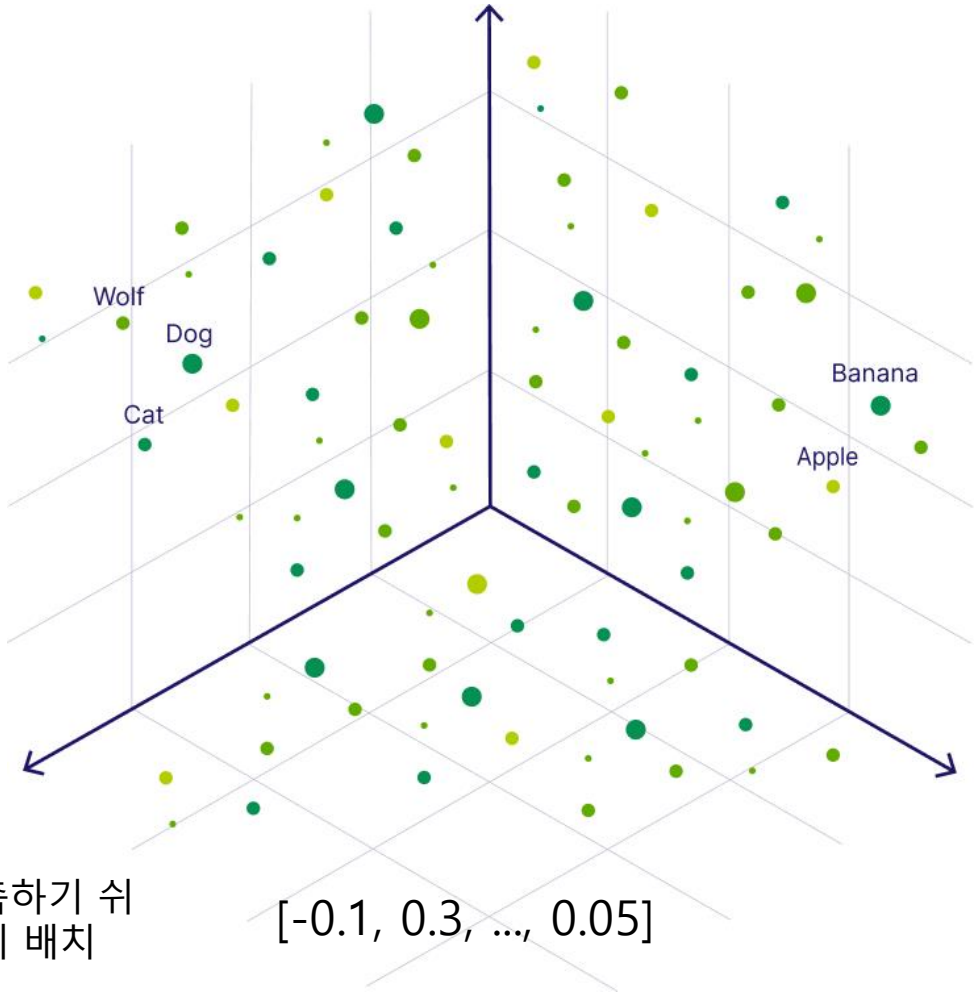
```
embedding = nn.Embedding(vocab_size, d_model)
token_vec = embedding(token_ids)
```

[-0.1, 0.3, ..., 0.05]



사전, 토큰, 임베딩

문장 → 토크나이징(tokenizing) → 토큰 ID → 임베딩 벡터



단어들을 "모델이 예측하기 쉬운 방식"으로 공간에 배치

[-0.1, 0.3, ..., 0.05]

cos_sim = F.cosine_similarity(embedding("apple"), embedding("banana"))

사전, 토큰, 임베딩

문장 → 토크나이징(tokenizing) → 토큰 ID → 임베딩 벡터

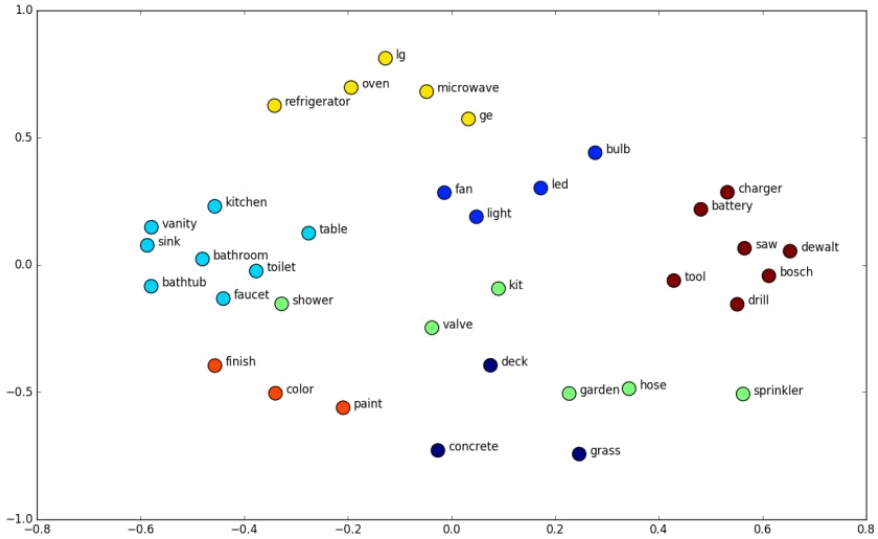
Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	Femininity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
Boy	0	1	0
Girl	1	1	0
Prince	0	1	1
Princess	1	1	1
Queen	1	0	1
King	0	0	1
Monarch	0.5	0.5	1

Each word gets a 1x3 vector

Similar words... similar vectors

Intro to Word Embeddings and Vectors for Text Analysis.

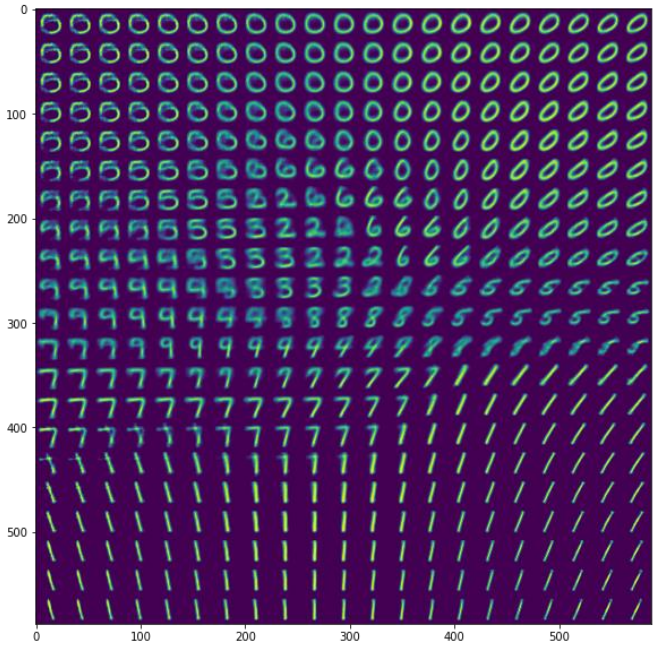


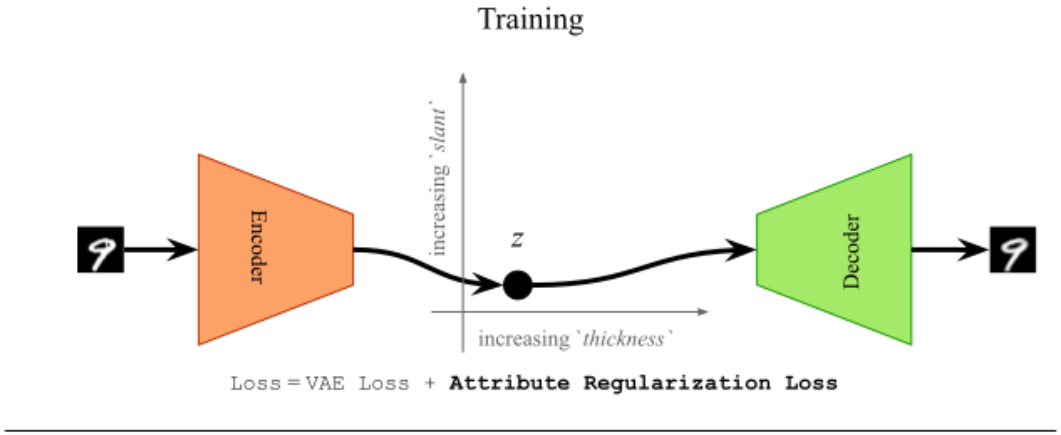
http://suriyadeepan.github.io/

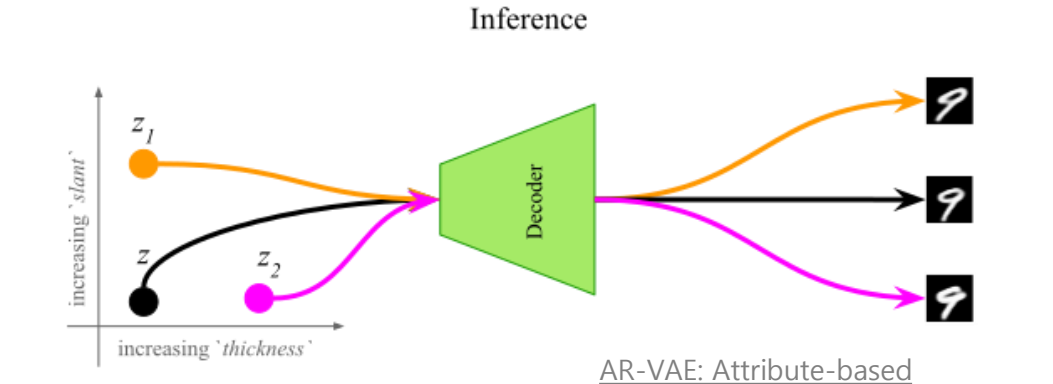
VAE

변이 자동 인코더(VAE. Variational autoencoder)

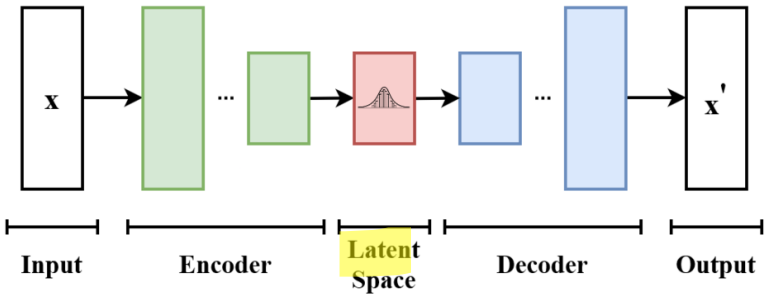
VAE는 압축된 잠재 공간에 데이터를 인코딩한 다음 다시 데이터로 디코딩하는 방법을 학습하는 생성 모델. 이는 특히 입력 데이터의 변형을 생성하는 데 유용.







AR-VAE: Attribute-based Regularization of VAE Latent Spaces | Music Informatics Group

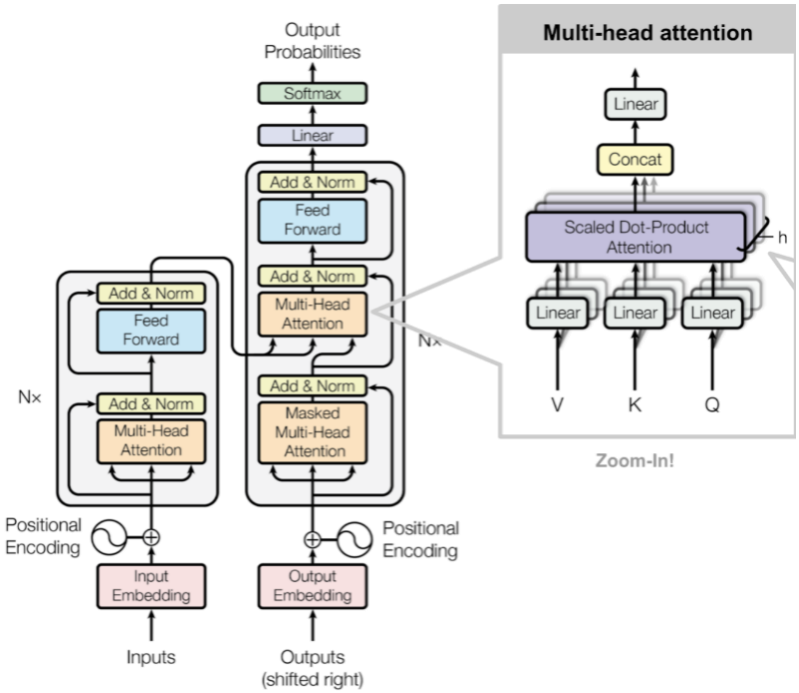


The Latent Space | Bogdan Penkovsky, PhD

Transformers

변환기(Transformers)

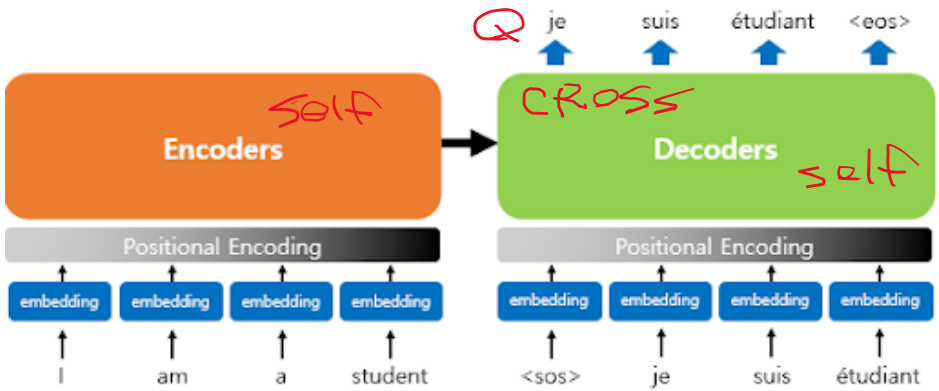
텍스트 생성 영역에서 GPT(Generative Pre-trained Transformer)와 같은 변환기 기반 모델은 방대한 양의 텍스트 데이터에서 어텐션(Attention)계산을 통해 패턴을 학습. 일관되고 맥락적으로 관련성 있는 토큰 시퀀스 학습 가능.
조건화된 이미지 생성에 사용.

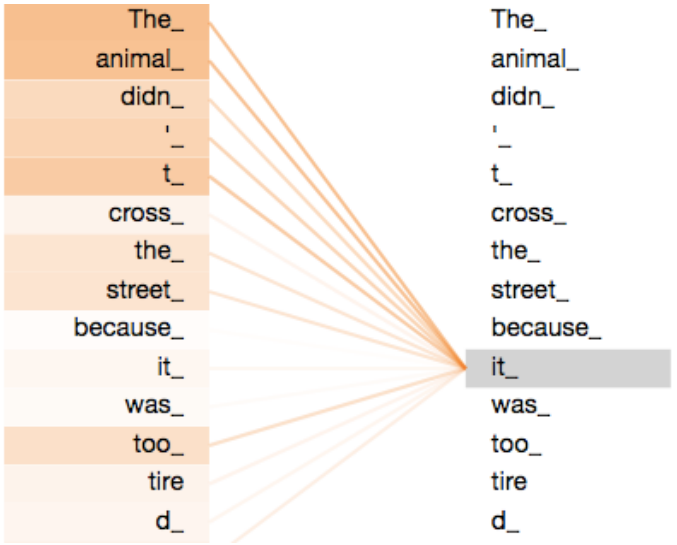


Five most popular similarity measures implementation in python - Dataaspirant

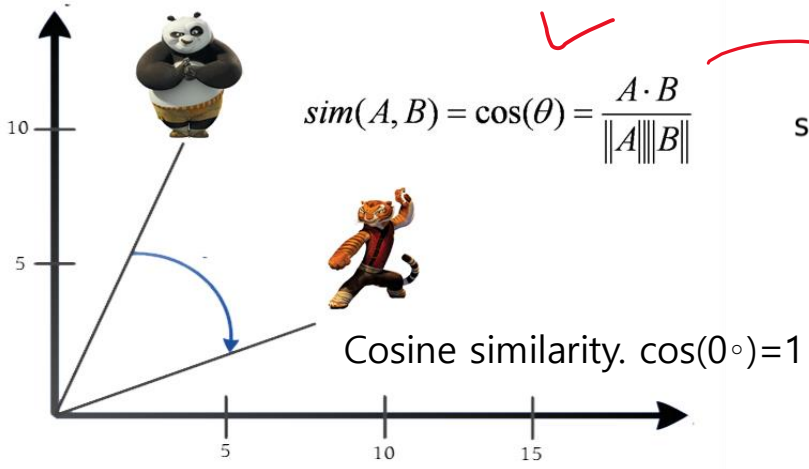
Daddy Makers: 딥러닝 모델 트랜스포머 인코더 핵심 코드 구현을 통한 동작 메커니즘 이해하기

Transformers





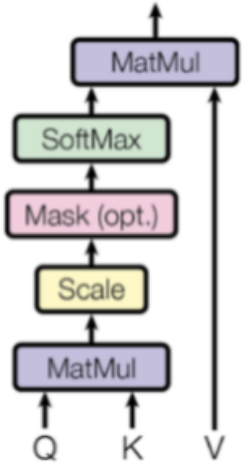
Five most popular similarity measures implementation in python - Dataaspirant



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

$$\mathbf{V} \times \text{Softmax}(\mathbf{Q}\mathbf{K}^T)$$

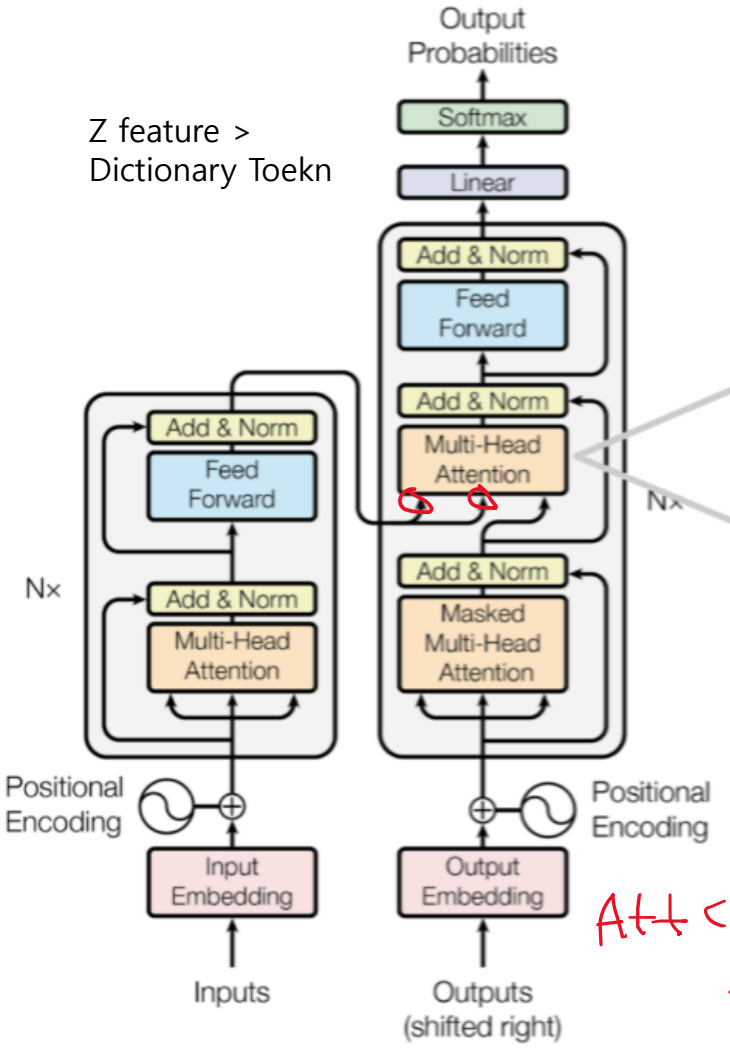


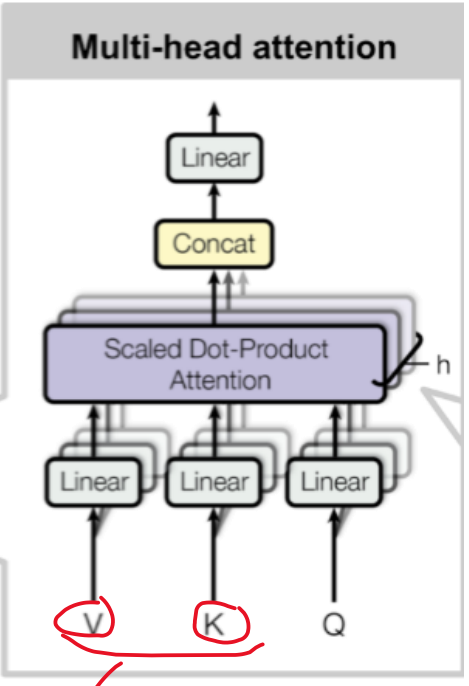
Five most popular similarity measures implementation in python - Dataaspirant

Daddy Makers: 딥러닝 모델 트랜스포머 인코더 핵심 코드 구현을 통한 동작 메커니즘 이해하기

Attention & Transformers

Z feature >
Dictionary Toekn

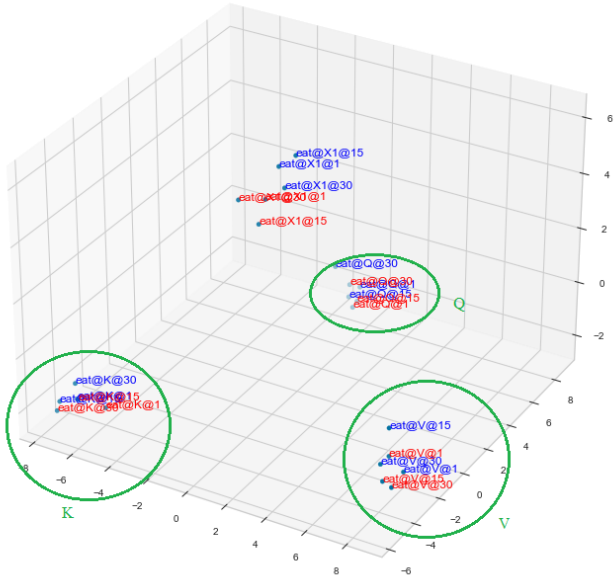




Zoom-In!

가중치가 포함된
토큰 시퀀스 맥락

Self attention
전체를 바라보며 문맥을 통합한
contextual embedding



Self Attention Space

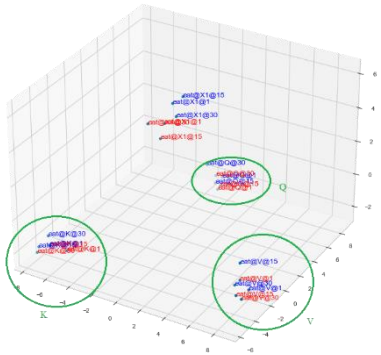
$$Att(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

ex) source code

GAN, VAE, Transformers

"나는 학생" → 어텐션 → out → FFN → logits (10000차원) → softmax → "입니다" (예측)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



out	문맥 반영된 벡터 (차원: `[seq_len, d_model]`)
Feed-Forward Network (FFN)	각 `out` 벡터를 비선형적으로 변환 (정보 강화)
디코더 스택 마지막 출력	최종 출력 hidden state
Linear + Softmax	각 hidden state에 대해 전체 vocabulary 크기만큼의 로짓(logits)을 만듦 → softmax를 적용해 확률로 변환
토큰 예측	확률이 가장 높은 단어가 다음 단어로 예측됨 (예: `"student"` 다음에 `"is"` 예측 등)

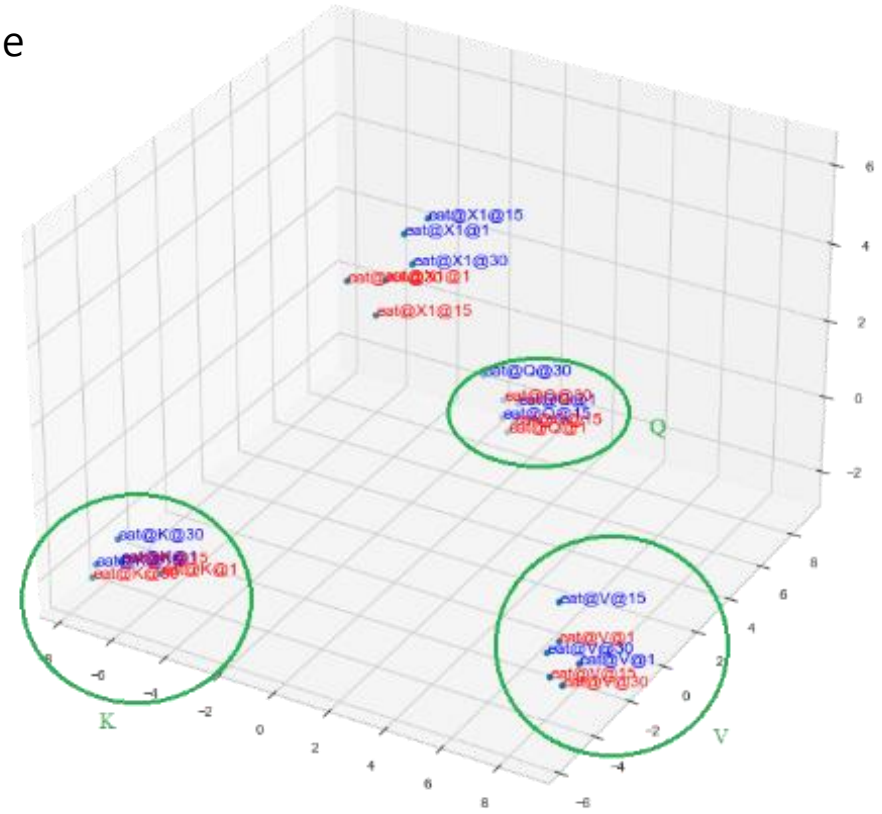
ex) source code

Transformers

- 처음엔 QK^T가 의미 없는 유사도를 계산함 → softmax 후 V를 평균해서 out 만듦
- 이 결과가 **예측 라벨(예: 다음 단어)**과 멀면 loss ↑
- 역전파로 Q, K, V를 만드는 가중치 W_Q, W_K, W_V가 업데이트됨
- 이 과정이 **수천만 문장을 반복하면서** 각 Q/K/V가 문맥에서 다른 역할을 하도록 학습됨

역할 = Query Key Value

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Transformers

```
# === 1. 입력 및 라벨 시퀀스 ===

input_sentence = ["I", "am", "a", "student"]          # 영어 입력
target_sentence = ["<sos>", "저는", "학생", "입니다"] # 한국어 출력 입력 (디코더 입력)
target_labels = ["저는", "학생", "입니다", "<eos>"]    # 예측할 실제 정답 (디코더 출력)


# === 2. 인코더 ===

X = embed(input_sentence)                             # 입력 시퀀스 임베딩 → [x_I, x_am, x_a, x_student]

Q_enc = linear_Q(X)                                    # Q from encoder input
K_enc = linear_K(X)                                   # K from encoder input
V_enc = linear_V(X)                                   # V from encoder input

encoder_outputs = self_attention(Q_enc, K_enc, V_enc) # 인코더 출력: 입력 문맥이 반영된 벡터들
```

Transformers

```
# === 3. 디코더 ===

Y = ["<sos>"]                                   # 디코더 초기 입력 (<sos> 부터 시작)
logits_sequence = []                            # 각 시점의 출력 로짓을 저장

for t in range(len(target_labels)):

    Y_embed = embed(Y)                          # 현재까지 생성된 디코더 입력 임베딩

    # --- 디코더 Self-Attention (마스킹 포함) ---
    Q_dec = linear_Q_dec(Y_embed)               # Q from decoder input
    K_dec = linear_K_dec(Y_embed)               # K from decoder input
    V_dec = linear_V_dec(Y_embed)               # V from decoder input
    dec_out = masked_self_attention(Q_dec, K_dec, V_dec) # 디코더 내부 문맥 계산 (미래 마스킹 포함)

    # --- Cross-Attention: 디코더 → 인코더 입력 참조 ---
    Q_cross = linear_Q_cross(dec_out)           # Q from 디코더 문맥
    K_cross = linear_K_cross(encoder_outputs)   # K from 인코더 출력
    V_cross = linear_V_cross(encoder_outputs)   # V from 인코더 출력
    cross_out = attention(Q_cross, K_cross, V_cross)  # 입력 문장 정보에 기반한 번역 벡터

    # --- 출력 로짓 생성 ---
    logits = linear_output(cross_out)           # vocab 크기의 예측 로짓
    logits_sequence.append(logits[-1])          # 마지막 위치 로짓만 저장 (가장 최근 토큰 예측)

    # --- 다음 디코더 입력을 위해 정답을 넣음 (teacher forcing) ---
    next_token = target_sentence[t + 1]         # 예: "저는" 다음은 "학생"
    Y.append(next_token)                        # 다음 루프를 위한 디코더 입력에 추가
```

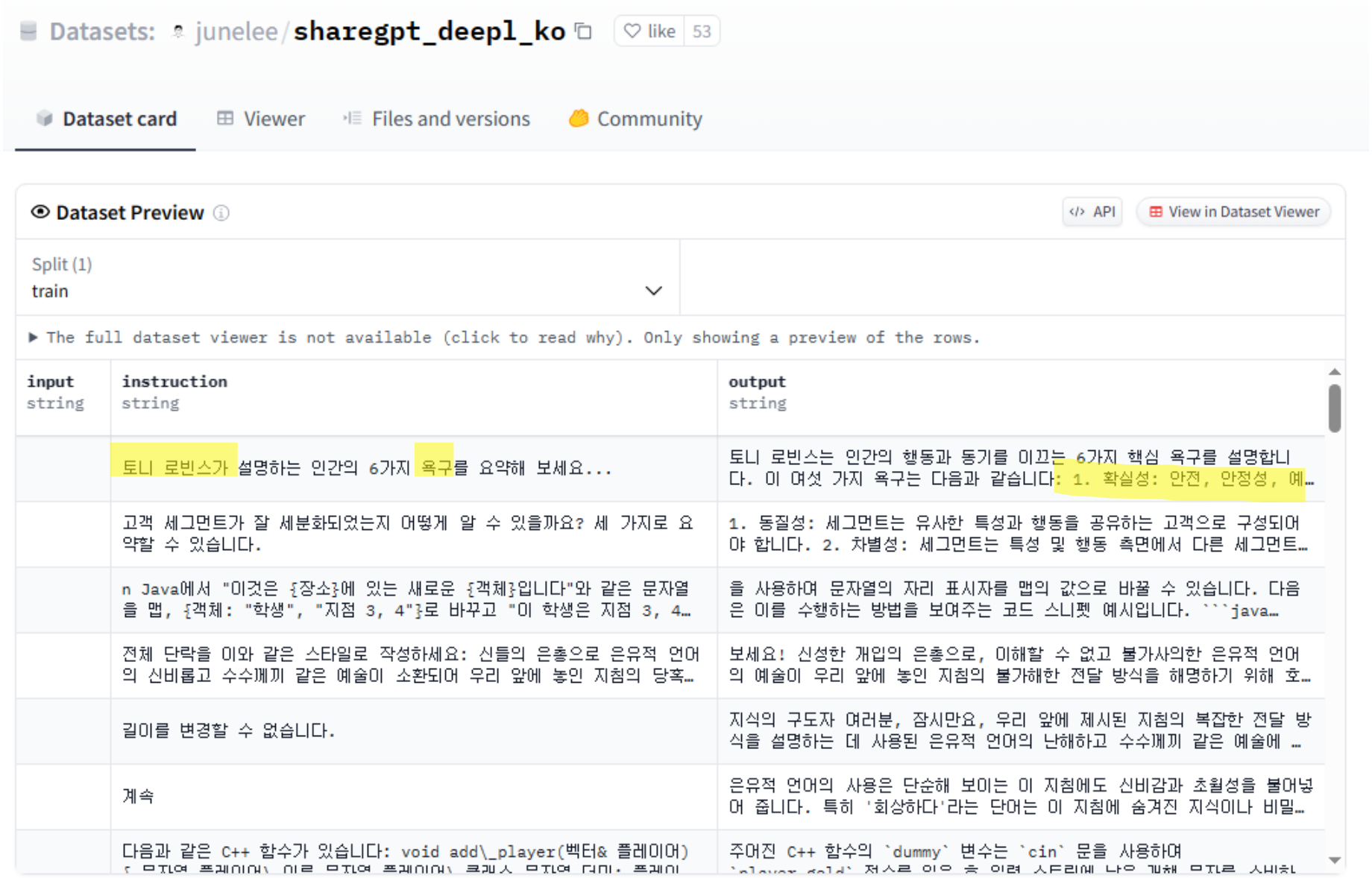
Transformers

```
# === 4. 손실(Loss) 계산 ===

loss = 0
for logit, label_token in zip(logits_sequence, target_labels):
    label_id = vocab_id(label_token)                  # 정답 토큰을 ID로 변환
    loss += cross_entropy_loss(logit, label_id)       # 예측 분포와 정답 ID 간의 CrossEntropy Loss

loss = loss / len(target_labels)                      # 평균 손실값 (시퀀스 길이만큼 나눔)

# === 역전파 및 파라미터 업데이트는 이 이후 단계 ===
```

Train and Dataset

instruction string · *lengths* 18 379	input string · *lengths* 0 1.42k	output string · *lengths* 10 5.93k
Find the most frequently used word in the given paragraph.	The cat is black, the dog is brown and the cow is white.	The: 3 times.
Generate a regex pattern to match the following string	String: "Hello123"	\w{5}[0-9]{3}
Write a code snippet to update the value of an element in an array.	Array: {3, 2, 5, 7, 8, 1} Index: 3 New value: 6	array[3] = 6;
Predict a label for the given sentence.	The phone is red.	Label: Color.
Print out the 5th element of an array.	arr = [55, 32, 10, 11, 8]	print(arr[4])
Classify the following text into two categories: 'Sports' and…	The critically acclaimed film 'Avatar' is about a paraplegic Marine who embark…	Entertainment
Create a formula in Excel to calculate the total cost given the price, tax,…	Price: 10 Tax: 0.08 Shipping: 6	=10+0.08*10+6
Classify the following given expression. Your response should be…	x = 0.5	float literal
Classify the following code as either a python function or code section.	x = "hello" z = x + " world"	Code section.
Generate a regex expression to match strings of length 8 with given…	Allowed letters: A-F and digits: 0-9	^[A-F0-9]{8}$

Train and Dataset



How to work and use it