医療とAIビッグデータ応用

AI総論

統合教育機構

須藤毅顕

AI総論

応用の演習で取り組んだこと

画像分類(MLP, CNN)や自然言語処理(Transformer, ChatGPT)

授業タイトル	授業内容
Google Colaboratoryの導入	Google Colaboratoryとは / Google Colaboratoryの基本操作
MNISTの読み込み	データセットの読み込みと画像データの理解
MNISTの可視化と加工	画像データの可視化と学習の前処理
MLP(Multi layer perceptron)①	MLPとは / 推論と学習の基本的概念
MLP②	MLPの推論と学習に関する基本演習
CNN(Convolutional Neural Network)	CNNの基本的概念と基本演習
データ拡張	画像処理におけるデータ拡張の基本的概念と基本演習
画像分類(グループ演習)	CIFAR10を用いたグループ演習
自然言語処理とChatGPT	自然言語処理の概論とChatGPTの導入
自然言語処理演習(NLP)①	Transformersのpipelineによる基本演習
自然言語処理演習②	TransformersのTokenizerによる基本演習
言語モデル-生成AI概論(講義)	言語モデルの基本的概念(講義)
教師なし機械学習-次元削減	次元削減の概論と基本演習
教師なし機械学習-クラスタリング	クラスタリングの概論と基本演習
AI総論(講義-生成AIや色々なAIモデル,社会実装他)	AIの応用例の紹介、AI総論(講義)

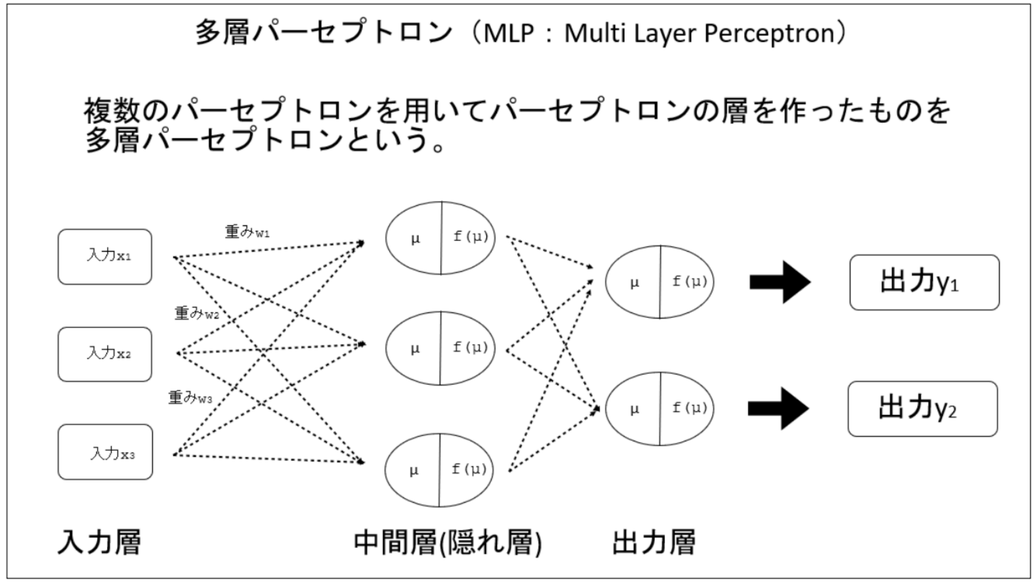
まだまだ深層学習は発展があります

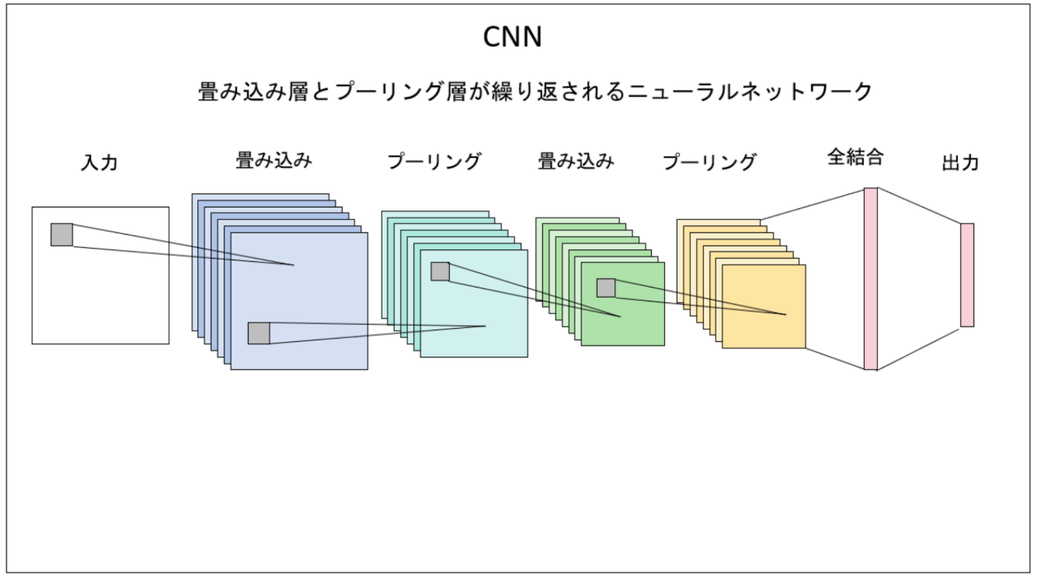
いくつか紹介していきます

画像認識(Computer Vision)

Computer Vision：コンピュータが画像やビデオから情報を抽出し解析するための技術や科学の分野

MLP、CNNでの画像分類(その一枚の画像が何かを分類)





画像分類は深層学習でのCVの代表例

数多くの代表的なネットワークが存在する(CNN(畳み込みとプーリング)が多い)

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010-2017年で実施された画像認識コンペ. 毎年上位入賞した手法は注目が集まる

ImageNetと呼ばれる1400万枚の画像が対象(1000クラス)

数多くの代表的なネットワークが存在する
(その多くは畳み込みとプーリング)

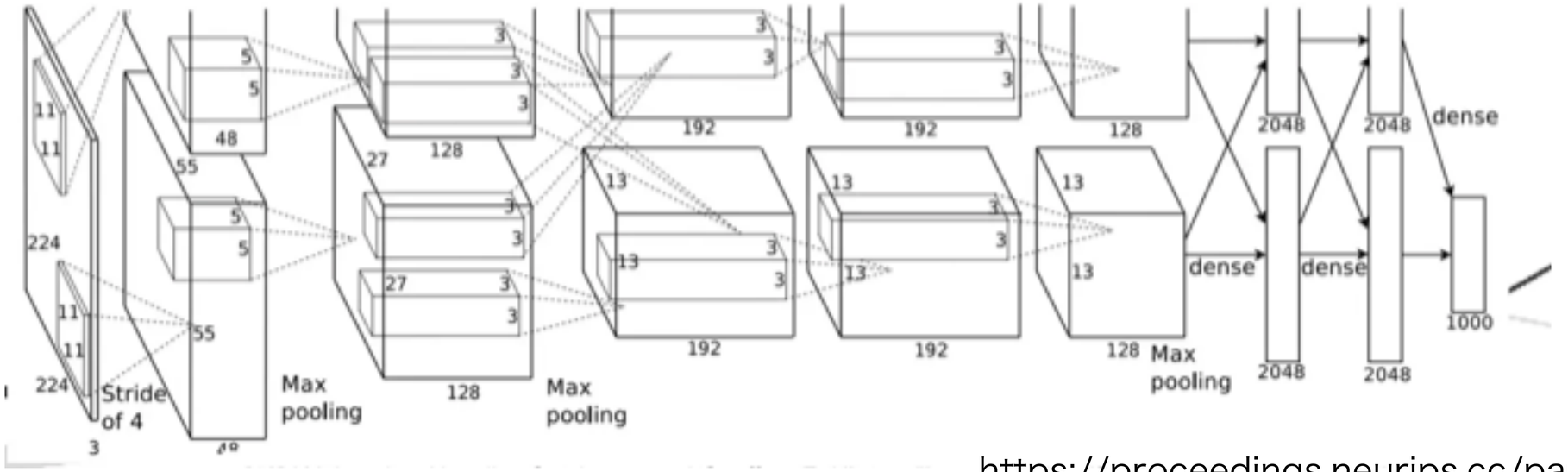
2012 : AlexNet
2014 : GoogleNet(1位), VGG (2位)
2015 : ResNet



ImageNet Large Scale Visual Recognition Challenge
Int J Comput Vis(2015) 115:211-252

AlexNet

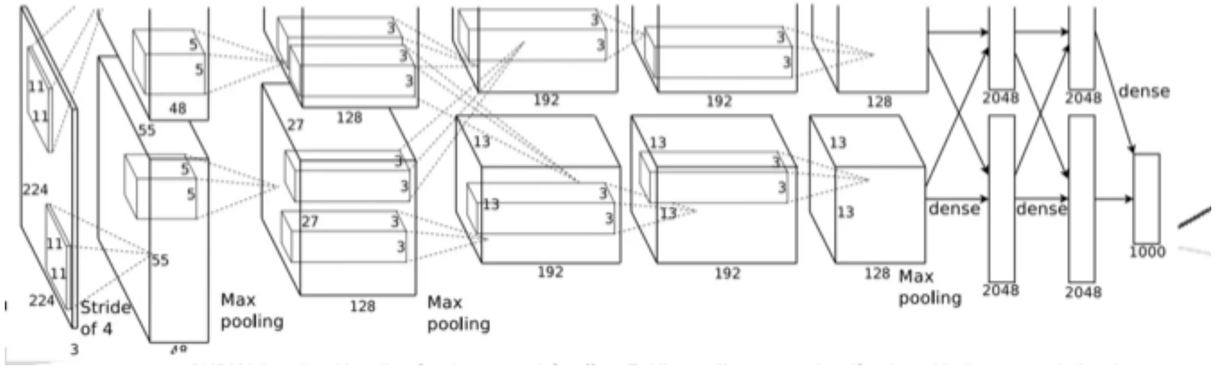
- 8層で構成
- ReLUを活性化関数に採用
- Dropout, Data Augmentationを採用



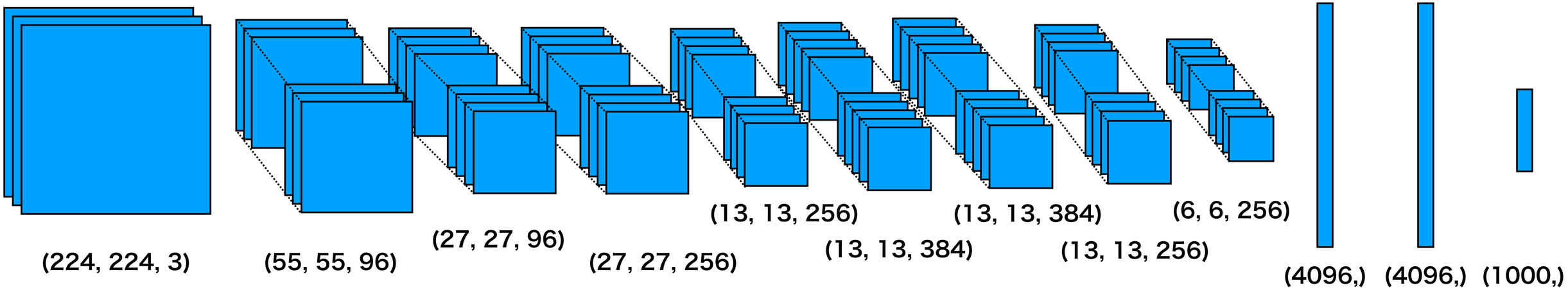
https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

基本的にCNNの構成

AlexNet

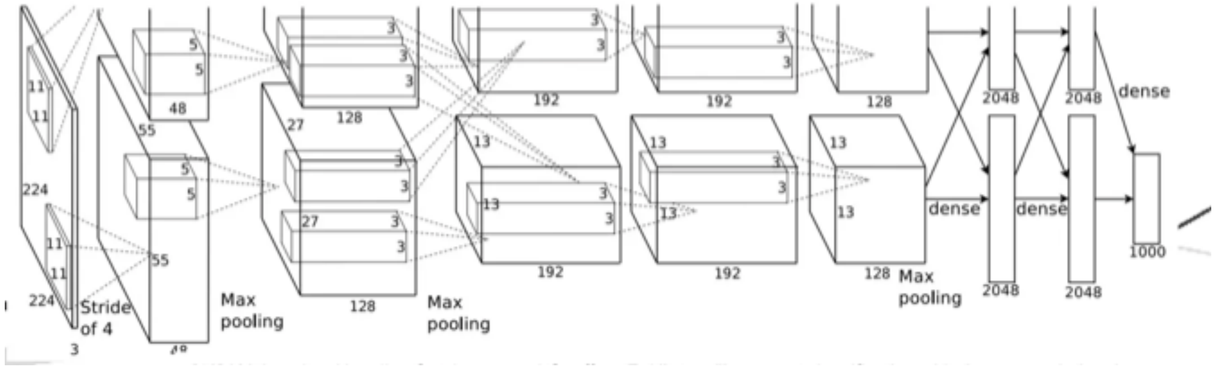


1 Conv　2 MaxPooling　2 Conv　2 MaxPooling　3 Conv　4 Conv　5 Conv　5 MaxPooling　6 全結合　7 全結合　8 全結合



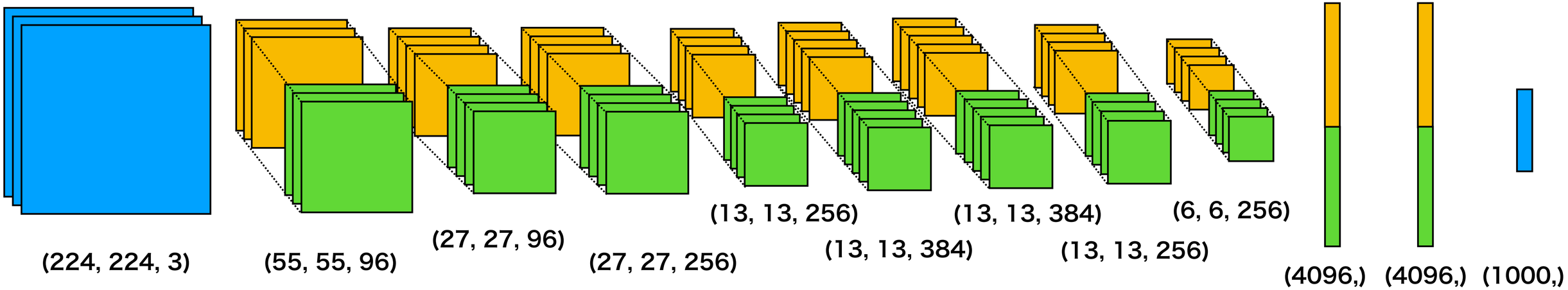
基本的にCNNの構成

AlexNet



分散処理
(2つのGPUで並列処理)

1		2		3	4	5		6	7	8
Conv	MaxPooling	Conv	MaxPooling	Conv	Conv	Conv	MaxPooling	全結合	全結合	全結合



基本的にCNNの構成

VGGNet

当時はまだ層が少ないのが当たり前だった中で急激に層を増やした

16層のVGG16と19層のVGG19が存在

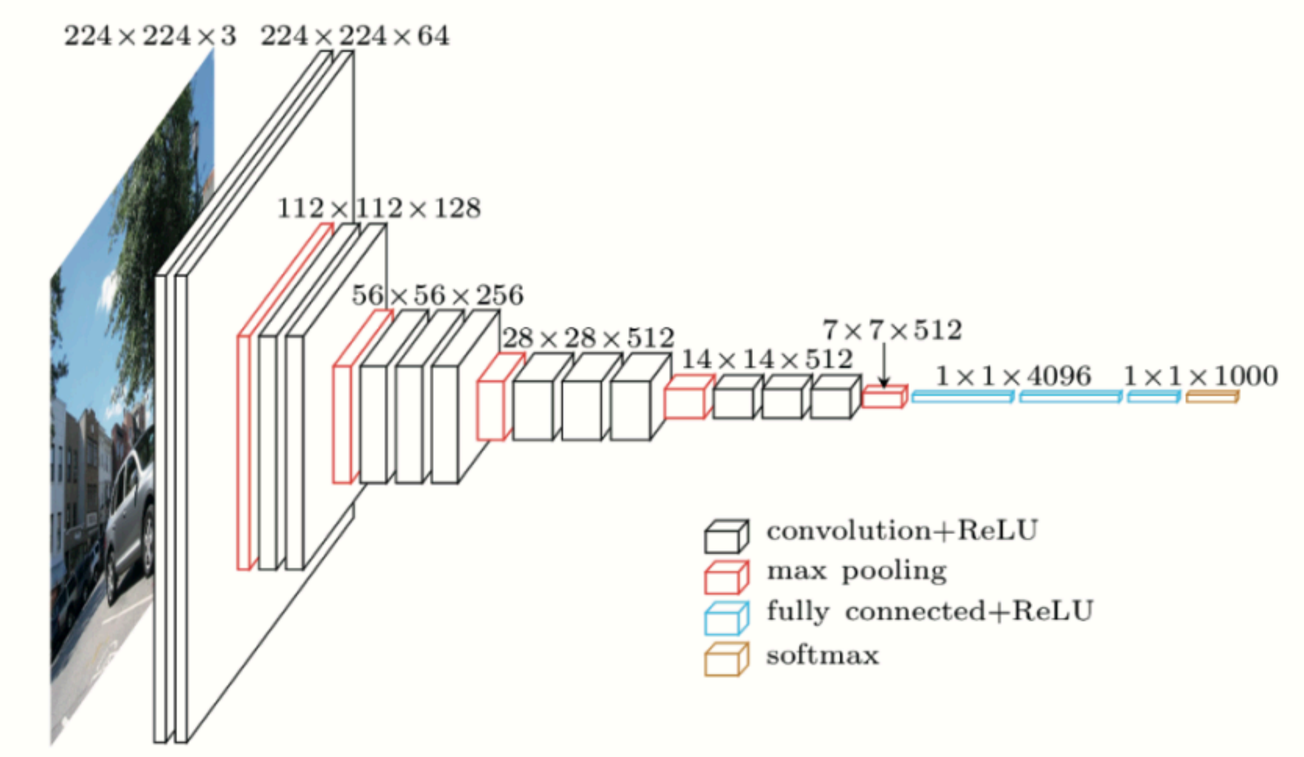


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

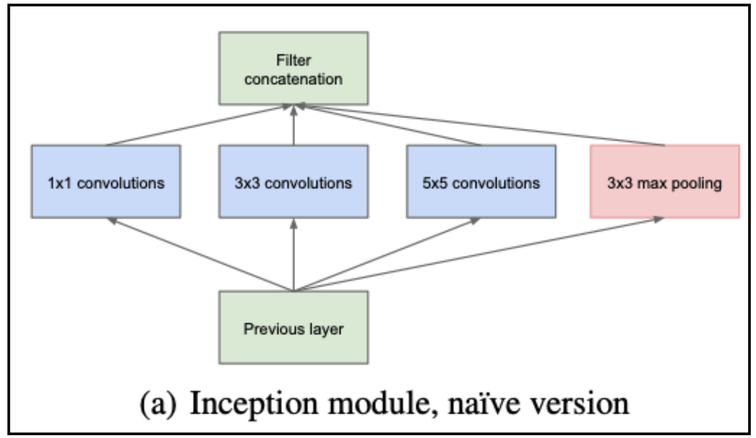
これも畳み込み、プーリング、全結合のシンプルなCNNの構造

GoogleNet　　ResNet

GoogleNet

Inceptionモジュールを導入
↑カーネルサイズが異なる畳み込み層を
並列に並べて結果を結合する

層の数は22層



(a) Inception module, naïve version

Christian Szegedy, et al.
arXive(2014)

ResNet

スキップ接続を導入
これにより層が増えると学習が難しくと言われていた中、152層で構成

複数のブロック間でスキップ接続するDenseNetも有名

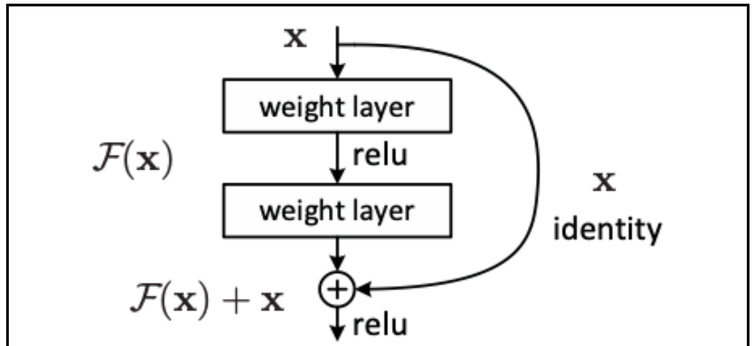


Figure 2. Residual learning: a building block.

Kaiming He, et al.
arXive(2015)

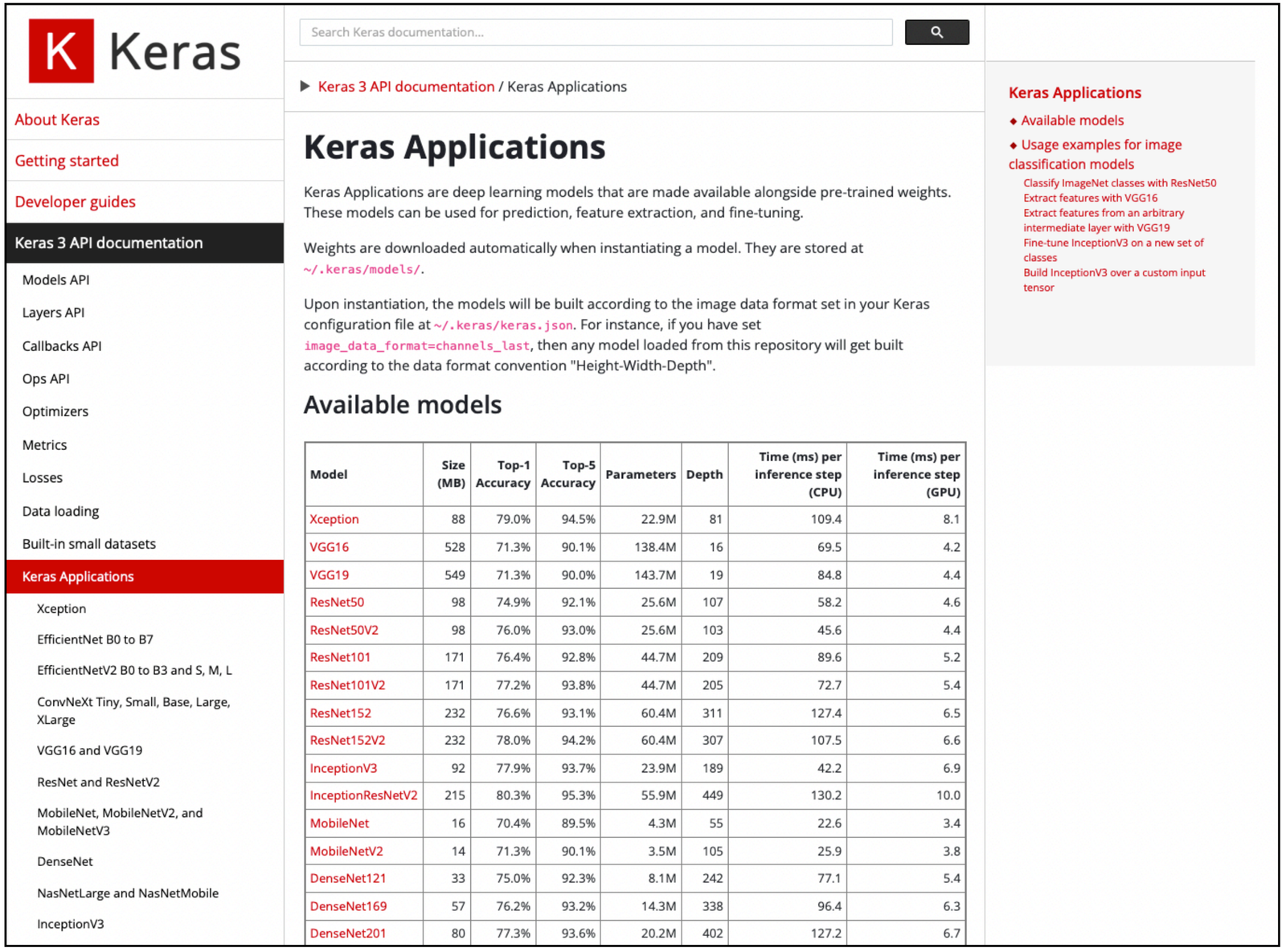
なぜこんな歴史を振り返るのか

現在も使用される

代表的な機構を知ることができる

転移学習に使用

新しい論文では過去のモデルと比較される



https://keras.io/api/applications/

簡単にモデルを実装できる

```
[2]  from tensorflow.keras.applications.vgg16 import VGG16

     model = VGG16()
     model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/ker
553467096/553467096 [==============================] - 22s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

そのまま予測につかう

一部を再学習(転移学習)

全てを再学習(ファインチューニング)

簡単にモデルを実装できる

```
[2]  from tensorflow.keras.applications.vgg16 import VGG16

     model = VGG16()
     model.summary()
```

そのまま予測につかう
一部を再学習(転移学習)
全てを再学習(ファインチューニング)

```
Downloading data from https://storage.googleapis.com/tensorflow/ker
553467096/553467096 [==============================] - 22s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from tensorflow.keras.preprocessing.image import load_img
dog = load_img("/content/drive/MyDrive/dog.png", target_size=(224,224))
```

dog



簡単にモデルを実装できる

```
[2]  from tensorflow.keras.applications.vgg16 import VGG16

     model = VGG16()
     model.summary()
```

そのまま予測につかう
一部を再学習(転移学習)
全てを再学習(ファインチューニング)

詳しい使い方はkeras公式HPを

```
Downloading data from https://storage.googleapis.com/tensorflow/ker
553467096/553467096 [==============================] - 22s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from tensorflow.keras.preprocessing.image import load_img
dog = load_img("/content/drive/MyDrive/dog.png", target_size=(224,224))
```

```
predictions = model.predict(inputs)
predictions

1/1 [==============================] - 1s 1s/step
array([[1.5987957e-06, 9.7035645e-06, 1.9131492e-07, ..., 1.1149690e-06,
        5.7272140e-05, 2.4959457e-04],
       [1.2654543e-06, 8.7922119e-05, 2.3226196e-06, ..., 2.3588773e-03,
        5.1664887e-04, 3.3934772e-05]], dtype=float32)
```

dog



VGG16には予測結果をクラス名に変換するdecode_predicitionがある.

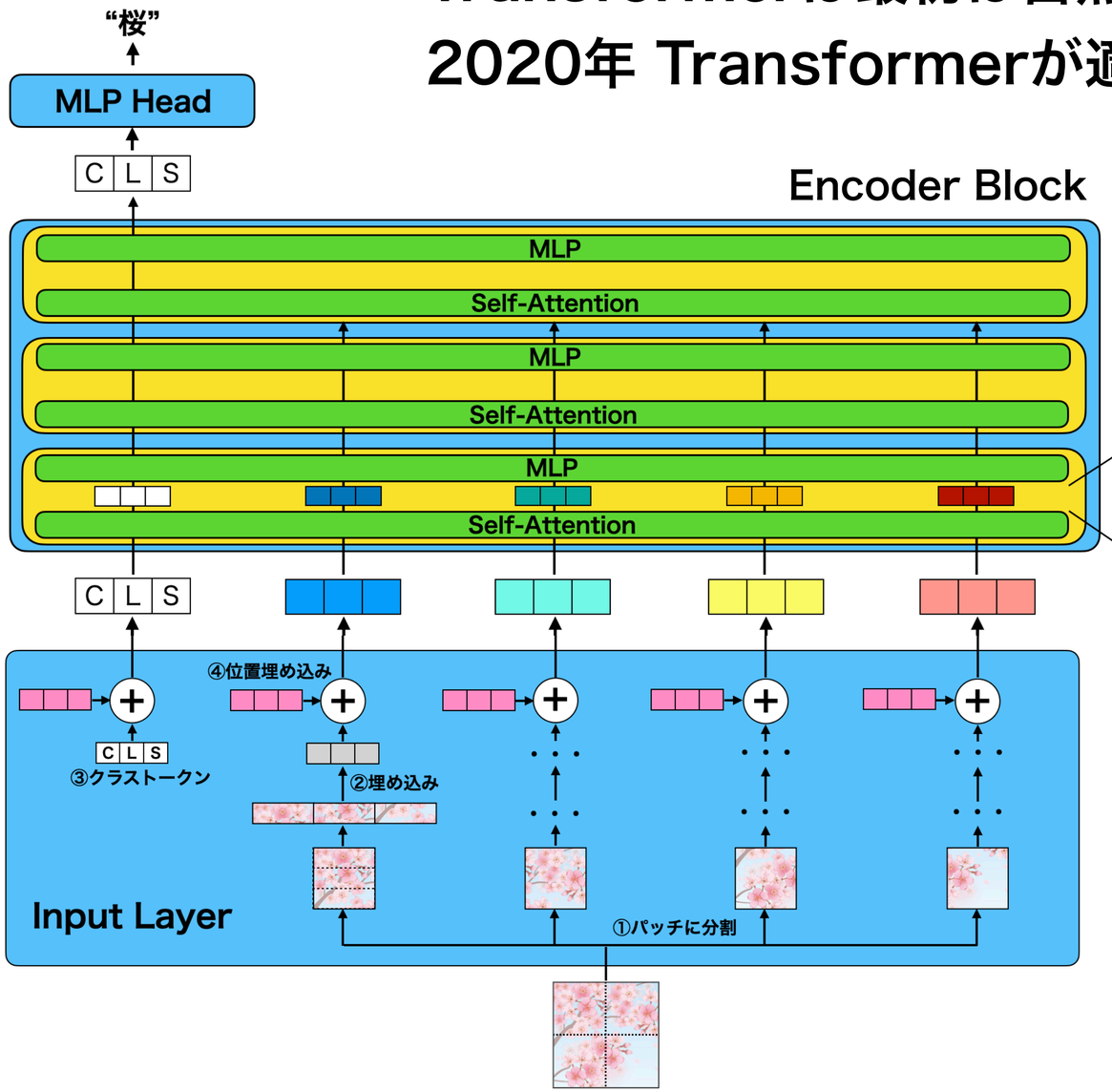
```
[17] from tensorflow.keras.applications.vgg16 import decode_predictions
     results = decode_predictions(predictions)
     # 結果はそれぞれの上位5位が出力される
     results

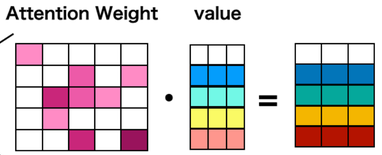
     Downloading data from https://storage.googleapis.com/download.tensorfl
     35363/35363 [==============================] - 0s 0us/step
     [[('n02099601', 'golden_retriever', 0.7427289),
       ('n02100877', 'Irish_setter', 0.09474462),
       ('n02090379', 'redbone', 0.027671015),
       ('n02099712', 'Labrador_retriever', 0.019016659),
       ('n02087394', 'Rhodesian_ridgeback', 0.015975796)],
```

予測上位5位を出力
←golden_retrieverが1位

Transformerの画像処理への応用：ViT

Transformerは最初は自然言語処理で応用
2020年 Transformerが適用されたViT(Vision Transformer)を発表





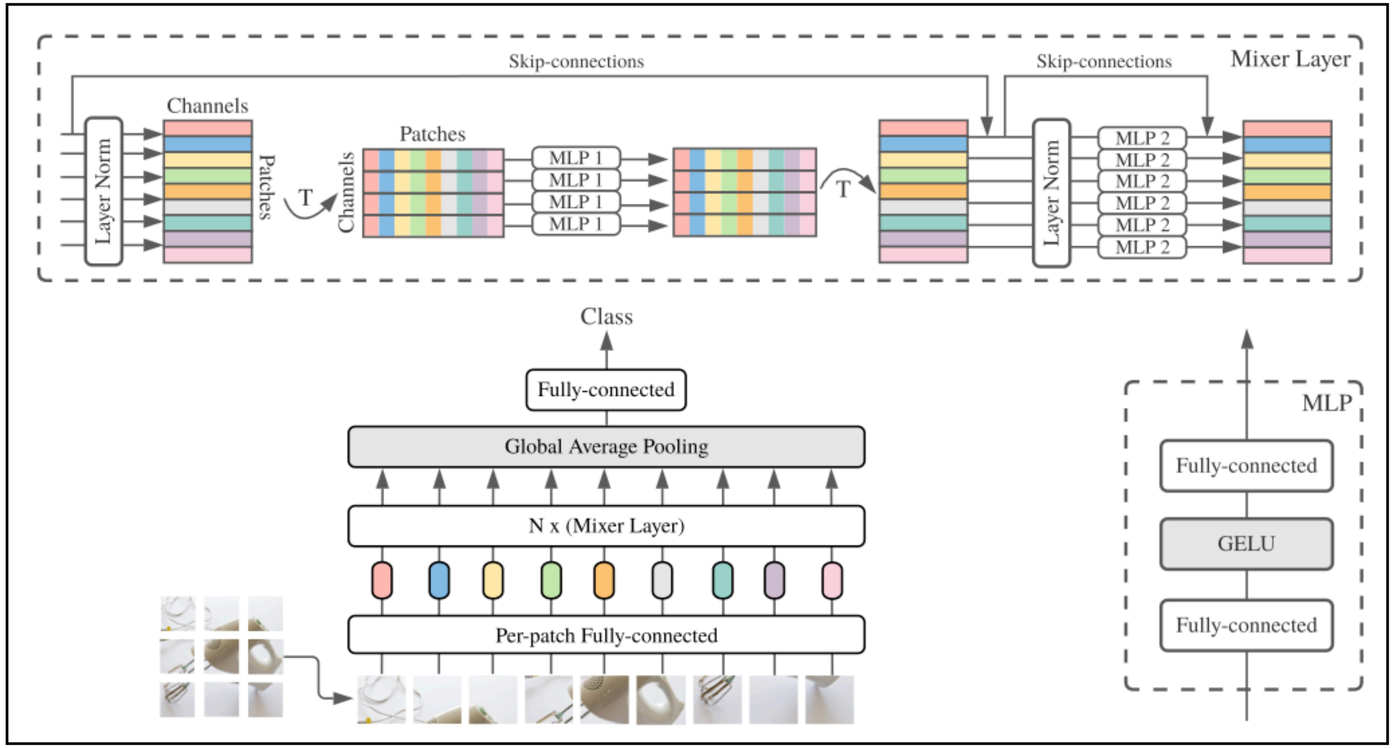
画像を16×16のパッチと呼ばれるサイズに分割して
256個の単語(トークン)のように扱う

３億枚の巨大なデータセット(JFT-300M)で
事前学習すると高い精度

CNNを使っていない

MLP復活？

MLP Mixer



Iiya Tolstikhin, et al.arXiv(2021)

- **Skip接続、パッチの分割などは行なっているがTransformerのアテンションやCNNは一切使っていない**
- **大規模データでの事前学習は必須**

画像分類

これまで行なってきたMLP、CNN、Transformerの
いずれも依然開発 or 研究が盛んに行われている

COVID19か否か

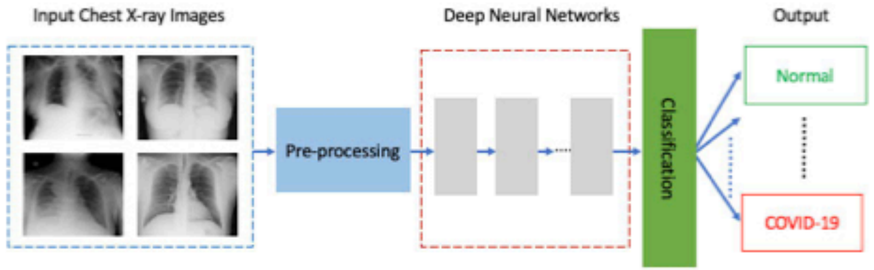


FIGURE 2. An overview of COVID-19 classification with a generic deep CNN.

S.S. Alahmari et al. IEEE Access.(2022)

虫歯か否か

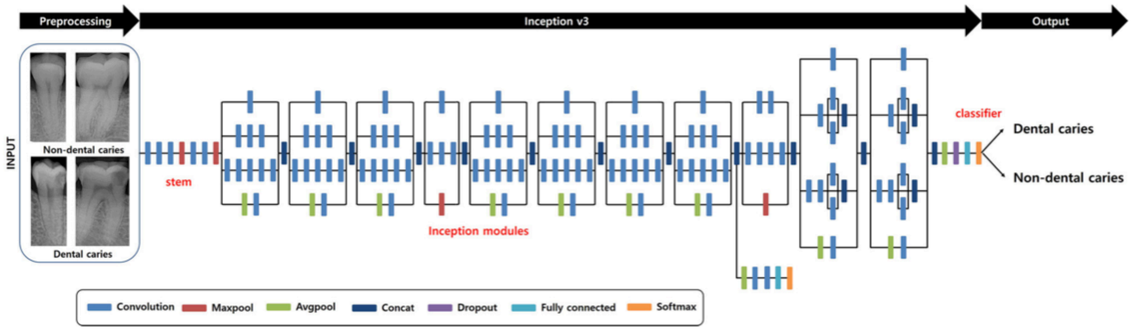


Fig. 1. Overall architecture of the pre-trained convolutional neural network model. The architecture has a total of 9 inception modules, including an auxiliary classifier, and two fully connected layers. The output layer performs binary classification between dental caries and non-dental caries using a softmax function.

J.H.Lee et al. Journal of Denstistry(2018)

その他の画像認識での代表的タスク：物体検出とセグメンテーション

画像分類以外の代表的なタスク：物体検出とセグメンテーション

画像分類：画像を一枚ずつ分類する



バスケをしている画像



サッカーをしている画像

画像分類以外の代表的なタスク：物体検出とセグメンテーション

物体検出：画像の中の物体を検出する



領域抽出(セグメンテーション)
：画像の中の物体の範囲を抽出する



物体検出

画像の中から物体の分類と位置を特定する

・バウンディングボックス検出



目的の物体を矩形(長方形)で囲むように特定

・キーポイント検出



検出したい物体の特定の点の位置を推定

物体検出にも多くのモデルが存在

1stageと2stageがある

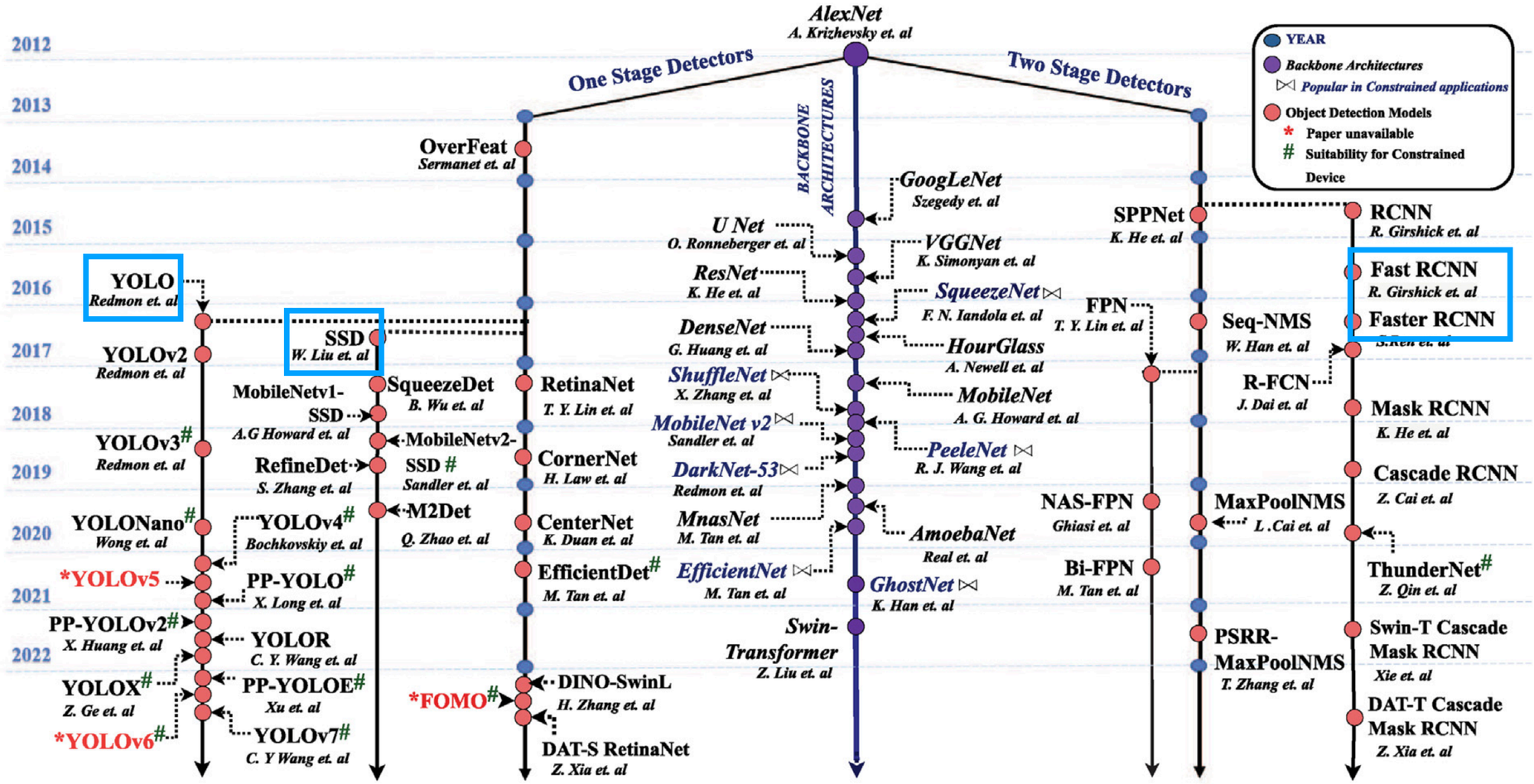
大雑把な違い

2stage
- 候補領域を抽出してからクラス分類を行う
- 1stageよりも高精度

ex) Fast R-CNN, Faster R-CNN

1stage
- 領域抽出とクラス分類を同時に行う
- 2stageよりも高速度

ex) SSD, YOLO



Vidya Kamath, et al.Neurocomputing(2023)

物体検出(YOLOv8)

ライブラリのインストール

```
!pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.1.5-py3-none-any.whl (702 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 702.8/702.8 kB 8.9 MB/
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/p
Requirement already satisfied: numpy>=1.22.2 in /usr/local/lib/pytho
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/li
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/pytho
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/pytho
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/py
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python
Requirement already satisfied: psutil in /usr/local/lib/python3.10/d
```

モデルを読み込んで推論

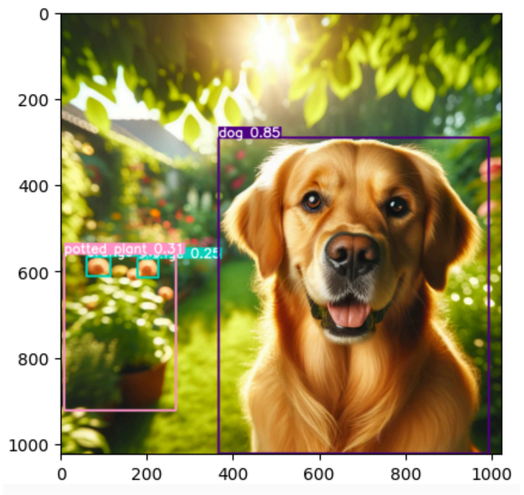
```
from ultralytics import YOLO
from PIL import Image
import matplotlib.pyplot as plt
import cv2

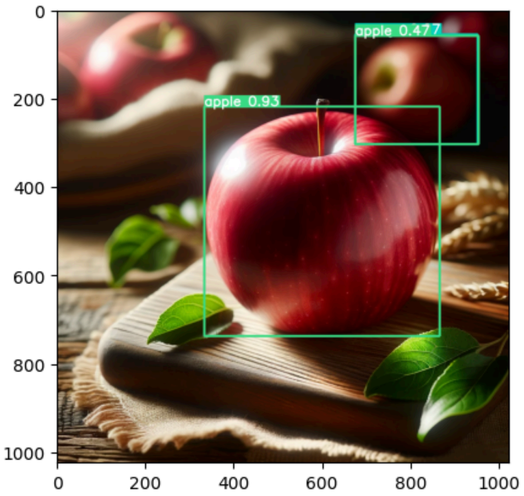
model = YOLO("yolov8n.pt")
img1 = "/content/drive/MyDrive/dog.png"
img2 = "/content/drive/MyDrive/apple.png"
results = model([img1,img2])
for r in results:
    im_array_bgr = r.plot()
    im_array = cv2.cvtColor(im_array_bgr, cv2.COLOR_BGR2RGB)  # plot a BGR numpy array of predictions
    plt.imshow(im_array)
    plt.show()
```



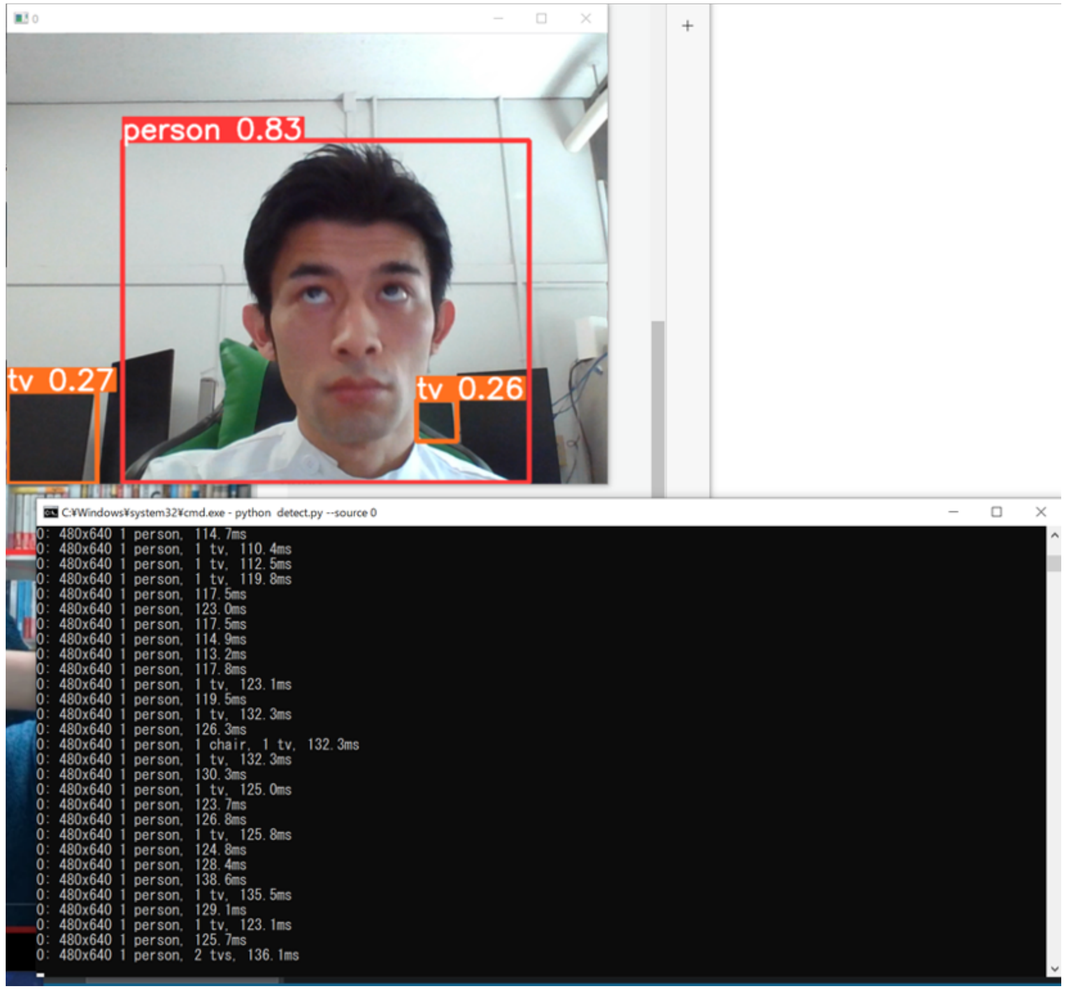


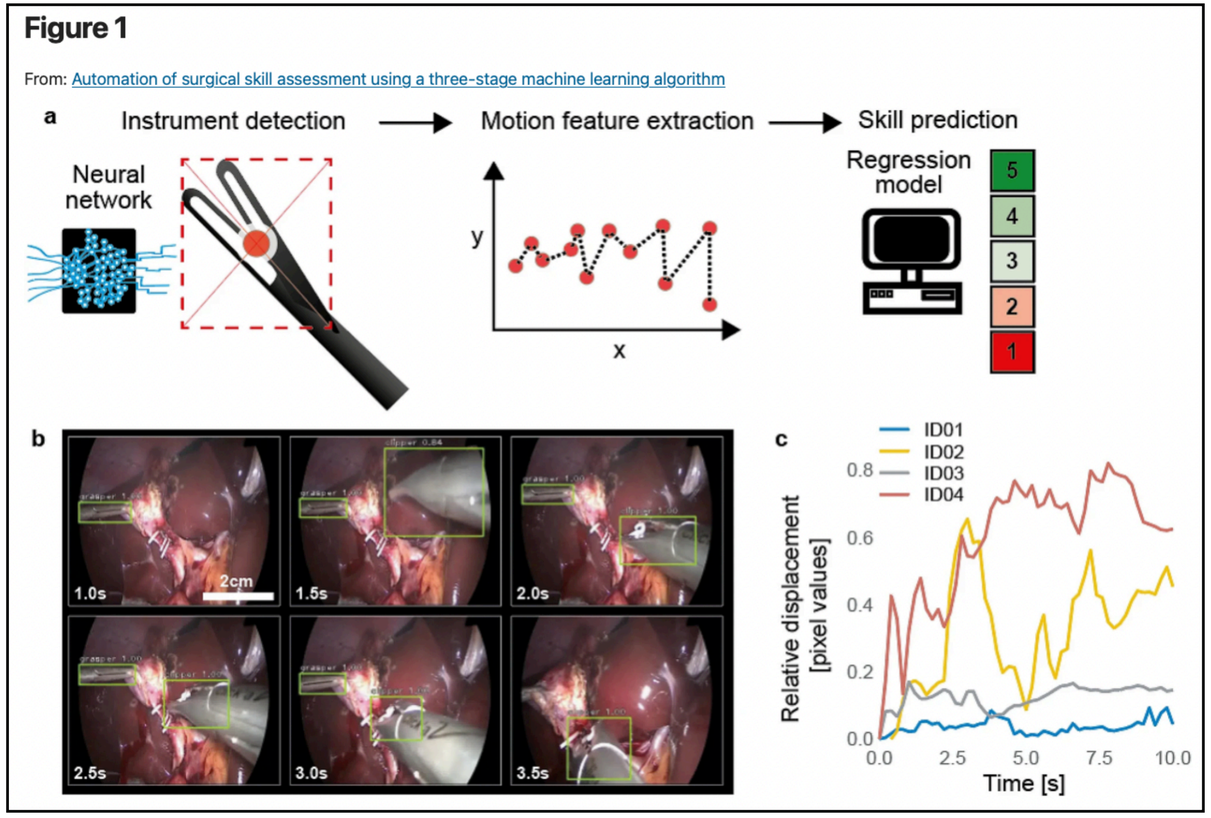






物体検出の推論速度↑　→ リアルタイム





Joel L. Lavanchy, et al. Scientific Reports(2021)

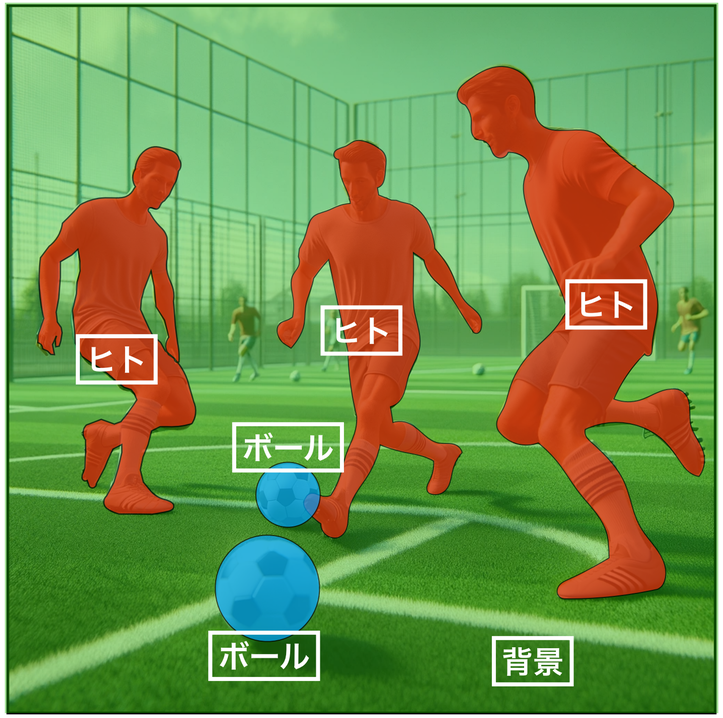
手術動画のリアルタイム物体検出

→術者の技量を予測

領域抽出(セグメンテーション)

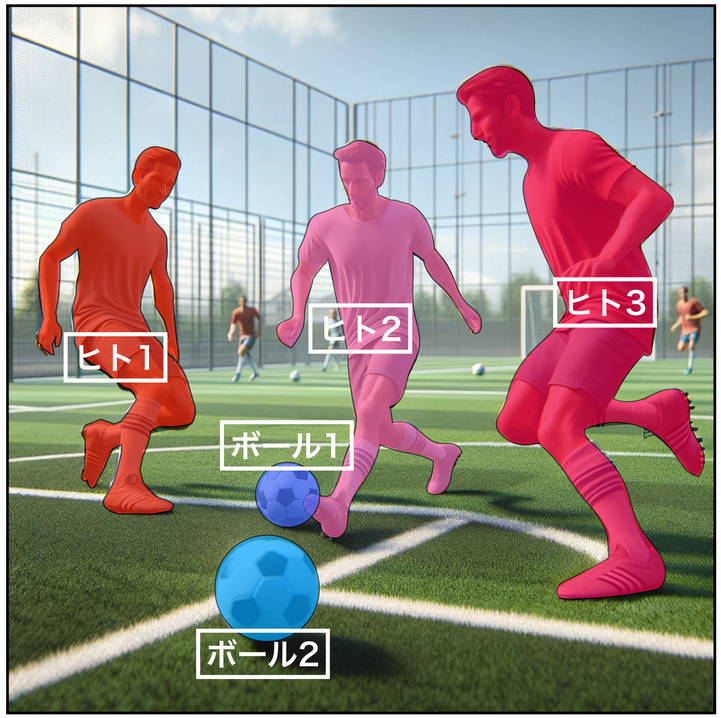
ピクセル単位でクラス分類を行う

セマンティック
セグメンテーション



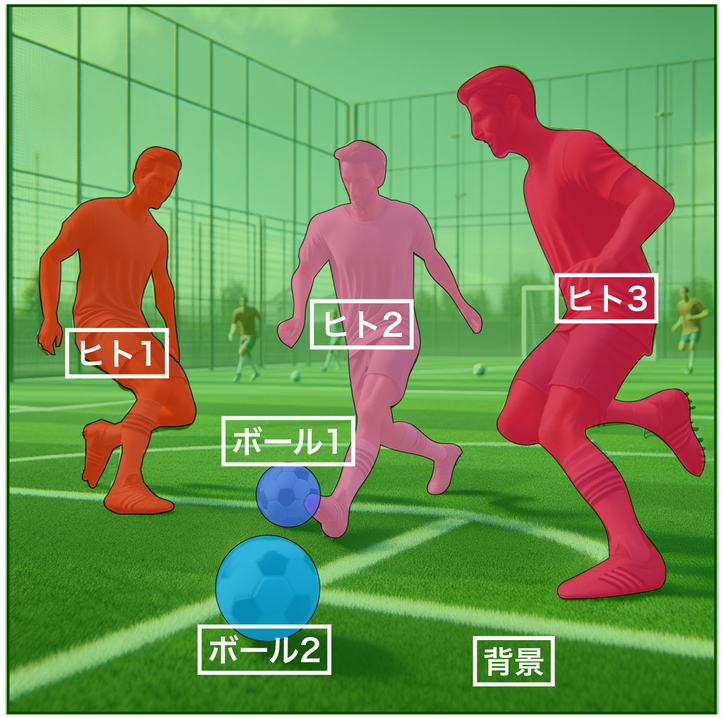
全てのピクセルに対してクラス分類

インスタンス
セグメンテーション



全ての物体に対してクラス分類

パノプティック
セグメンテーション



セマンティックセグメンテーション
＋
インスタンスセグメンテーション

セマンティックセグメンテーション(YOLOv8)

```python
from ultralytics import YOLO

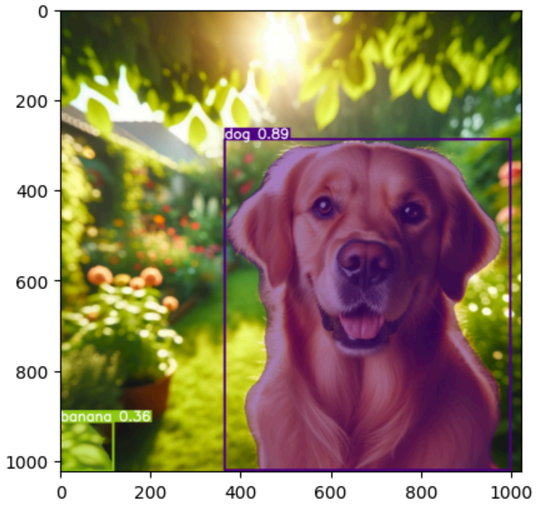
# Load a model
model2 = YOLO('yolov8n-seg.pt')  # load an official model
img1 = "/content/drive/MyDrive/dog.png"
img2 = "/content/drive/MyDrive/apple.png"
# Predict with the model
results2 = model2([img1,img2])
```

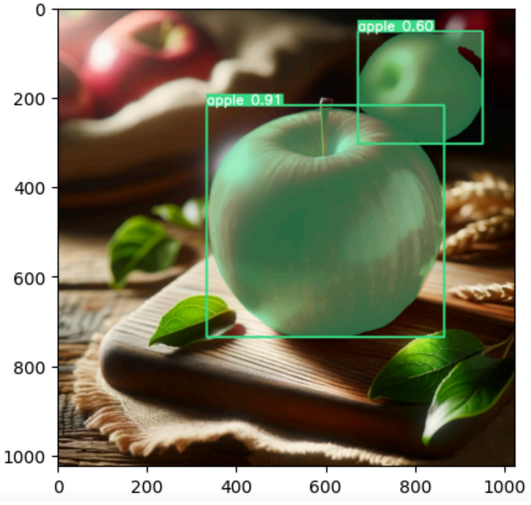
```python
for r in results2:
    im_array_bgr = r.plot()
    im_array = cv2.cvtColor(im_array_bgr, cv2.COLOR_BGR2RGB)
    plt.imshow(im_array)
    plt.show()
```









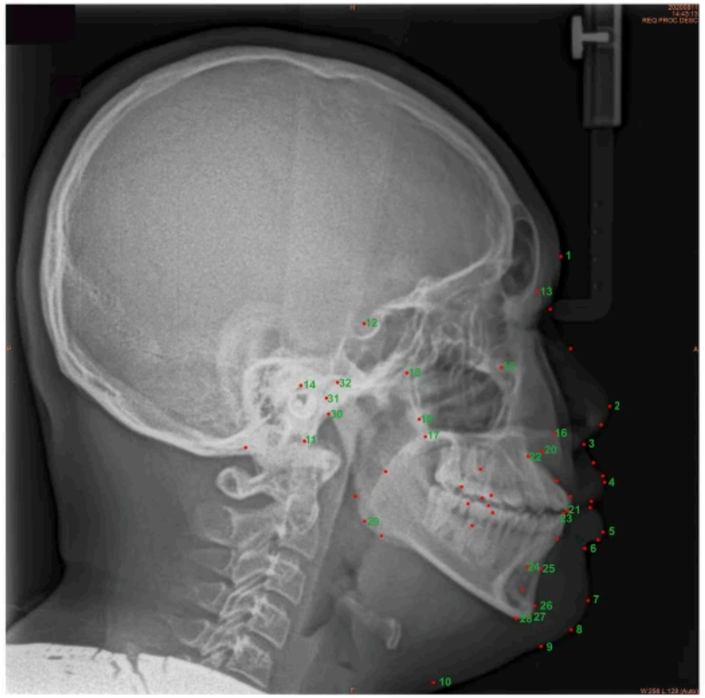


物体検出とセグメンテーション

セファログラム(矯正歯科)でのキーポイント検出

Fig. 1

The 32 anatomical landmarks used in this challenge



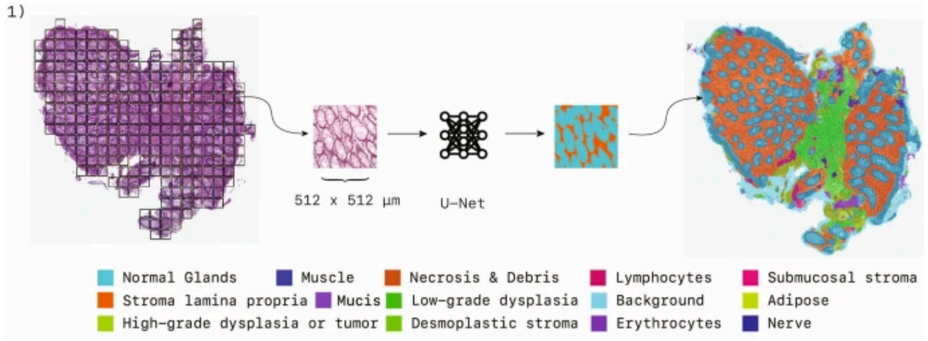
No.	Landmark
1	G
2	Prn
3	Sn
4	Ls
5	Li
6	Si
7	Pog'
8	Gn'
9	Me'
10	C
11	Ba
12	S
13	N'
14	P
15	Or
16	ANS
17	PNS
18	Pt
19	Ptm
20	A
21	U1
22	UIA
23	C1
24	LIA
25	B
26	Pog
27	Gn
28	Me
29	Go
30	Ar
31	Pcd
32	Co

The 32 anatomical landmarks used in this challenge. All landmarks are defined and explained in Table 1

Huayu Ye, et al. BMC Oral Health(2023)

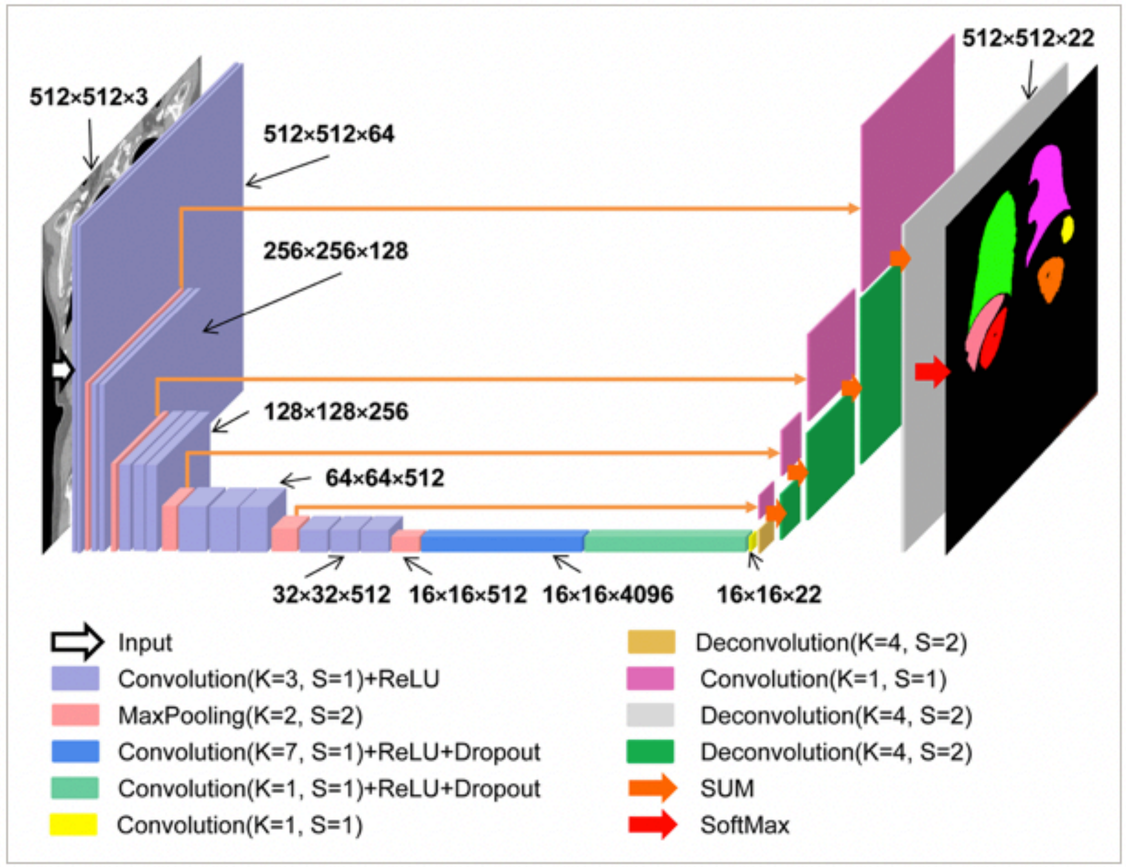
切片(大腸がん)のセグメンテーション

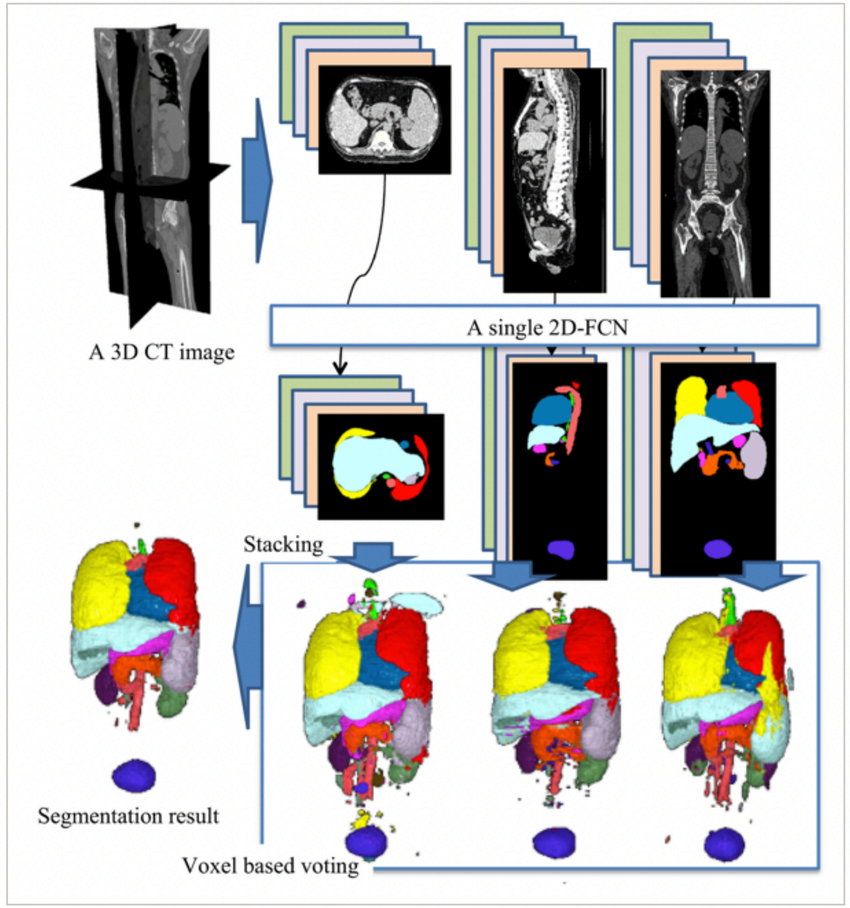
Figure 5



J M Bokhorst, et al. Scientific Report(2023)

3次元画像





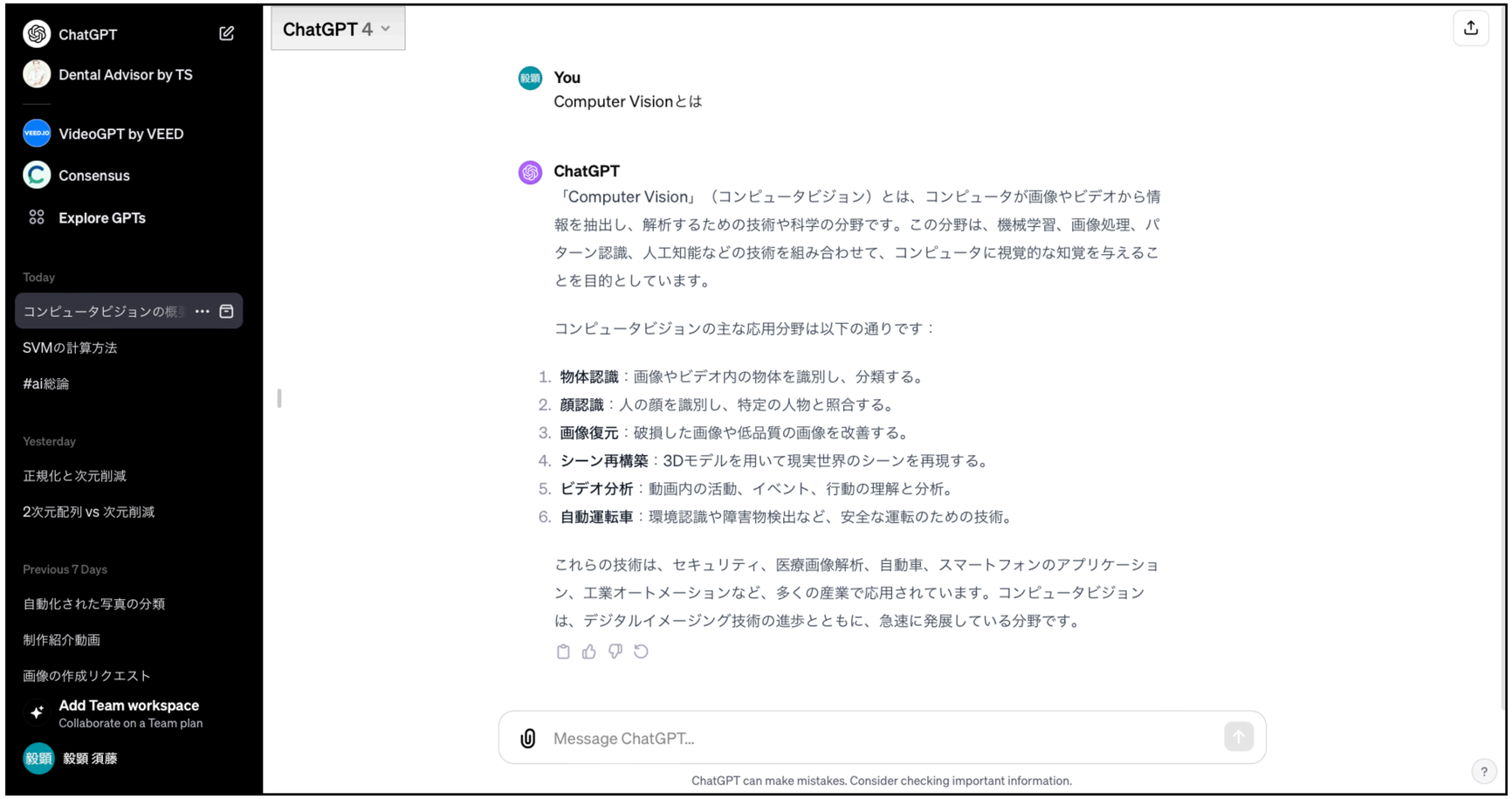
Xiangrong Zhou, et al. MEDICAL PHYSICS(2017)

CT画像をセグメンテーションした後に3次元画像として構築

応用後半の自然言語処理&生成AIについて

生成AIについて

ChatGPTで急速にAIが身近になった



強烈な汎用AI

優れたUI

生成AI ≠ ChatGPTではない

生成AIとその応用

ChatGPTは生成AIの1つのアプリケーションに過ぎない

代表的なサービス

生成AI

- テキスト生成AI
  - ChatGPT
  - Bing AI
  - Bard
  - Notion AI
  - Claude2
- 画像生成AI
  - Stable Diffusion
  - DALL-E
  - Midjourney
- 音楽生成AI
  - MusicLM
  - MugicGen
  - WavTool
- 動画生成AI
  - Imagen Video
  - Make-A-Video
  - Gen-2
- UI/Webサイト生成AI
  - Galileo AI
  - Uizard

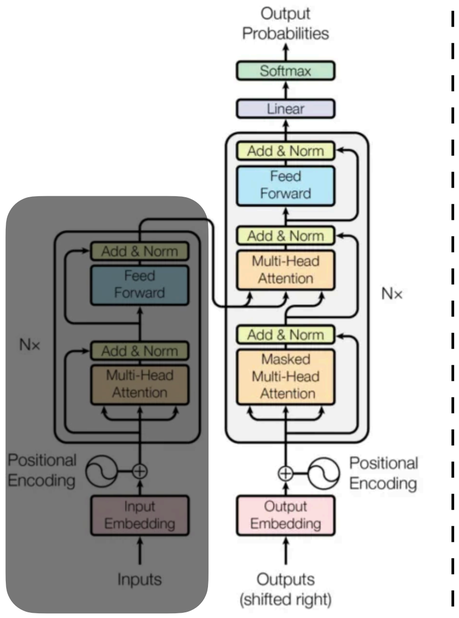
参考文献4)より

ChatGPTとGPT3.5, GPT4の関係

GPT3.5, GPT4：大規模言語モデル

大量の
テキスト

Transformer

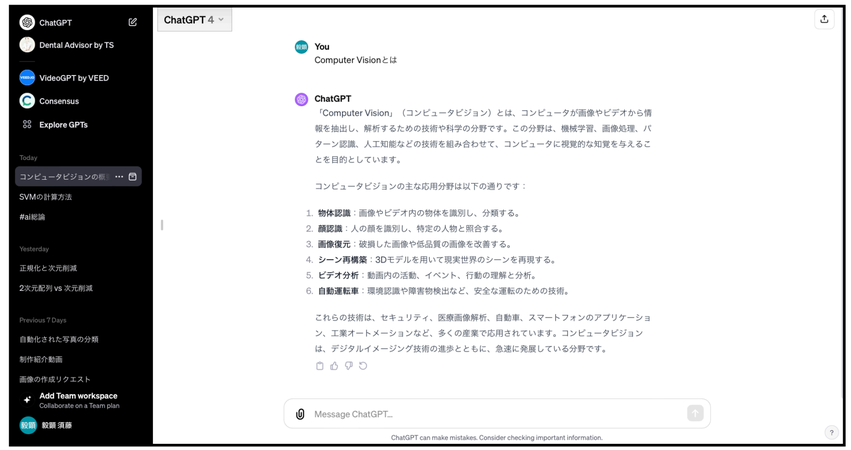


decoder部分を改変

アプリとして
使いやすく実装

(不適切な表現に
ならないように
さらに微調整)

ChatGPT：対話型アプリケーション



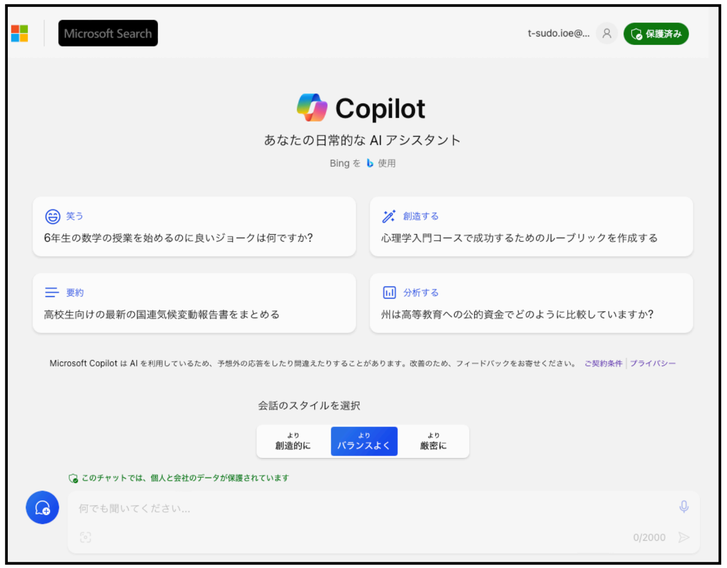
これらの生成AIは基本的にテキスト→テキスト

ChatGPTのライバル達

Bing、Bard、Claude2

LLM:大規模言語モデル

BingAI (Copilot)



LLM：GPT4

会社：Microsoft

Bard



LLM：LaMDA, PaLM2

会社：Google

Claude2



LLM：Claude2.1

会社：Anthoropic

これらはテキスト生成のチャットサービス

ChatGPTなどのテキスト生成AIで出来ること

出来ること	用途	プロンプトの例
質問に答える	一般的な知識問題に答える	「フランスの首都は何ですか？」
テキスト生成	クリエイティブな執筆やアイデアの提供	「ドラゴンと失われた王国についての物語を書いてください。」
画像生成	説明を基にしたビジュアルコンテンツの生成	「未来都市の夕日の画像を作成してください。」
ファイルブラウジング	アップロードされた文書の検索と分析	「教科書の「光合成」についてのセクションを探してください。」
Pythonコーディング	コードの実行、エラー修正、データ分析	「10の階乗を計算してください。」
ウェブブラウジング	情報の調査、要約、最新のデータやニュースの収集	「再生可能エネルギー源に関する最新の研究を見つけてください。」
翻訳	テキストの異なる言語への翻訳	「この文章を英語に翻訳してください。」
要約	長いテキストや記事の簡潔な要約	「このニュース記事を要約してください。」
文章の校正	文法やスタイルのチェック	「このエッセイの文法を校正してください。」
文体の変更	異なるトーンや文体での書き換え	「このパラグラフをカジュアルな口語文に書き換えてください。」
フォーマット整形	特定のフォーマットへの文書の整形	「この履歴書をビジネススタイルに整形してください。」
プログラミング解説	コードの概念の説明	「このPython関数がどのように動作するか説明してください。」
計算実行	複雑な数学的な計算	「この数列の合計を計算してください。」
教育支援	学習資料の提供や教育的なガイドラインの提案	「光合成のプロセスについて簡単に説明してください。」
心理学的サポート	一般的な心理学的アドバイス	「ストレスを管理する方法についてのアドバイスをください。」
健康アドバイス	一般的な健康情報の提供	「健康的な食生活についてのヒントを教えてください。」
表の作成	データの整理と視覚化	「売上データから四半期ごとの売上成績表を作成してください。」
画像認識	画像の内容分析と説明	「この画像に写っている物体を識別してください。」
データ分析	統計データの分析やビジュアル化	「このデータセットからトレンドを分析してください。」
プログラミング学習支援	プログラミング概念の説明や学習資料の提供	「オブジェクト指向プログラミングとは何ですか？」

赤は有料版(GPT4のみ)

基本的に使用者に対して優しい対応をするように調整されている

GAFA　　Meta(旧facebook)

2022/11/15(ChatGPT公開の2週後)対話型AI「Galactica」のデモをWebサイトで公開
→ヘイト表現など不適切な回答でSNSで炎上→3日で公開中止

(2022/8月に「BlenderBot3」(チャットボット)を公開したが、差別発言などで炎上)

「Galactica」は一般公開は中止したが、システムに対するフィードバックを得ることが短所を克服する最善の方法であるとして、研究者向けには公開を継続

参考文献4)より

Anthoropic

生成AIスタートアップ企業
2021年にOpenAIの元従業員6人で起業
CEOのアモディ兄弟はOpenAIで安全性などの研究に従事
Googleが約3億ドル出資し、株式約10%を取得



2023/3月　「Claude」を一般公開
→ヘイト表現などを生成する可能性が低いという評価
2023/5月　最大入力可能トークン数9000トークン→10万トークンに拡張
10万トークン=約7万5000字の英語の文章に相当
(GPT-4では最大3万2768トークン)
一般的な人間は10万トークンの文章を読むのに約5時間ほど→Claudeは1分以内

参考文献4)より

オープンソースのLLM

モデル名	開発元	パラメータ数	商用利用
HuggingChat	HuggingFace	300億	不可
StableLM	Stability AI	30億、70億、(150億、650億)	可
Dolly 2.0	Databricks	120億	可
MPT	MosaicML	70億、300億	可
BLOOM	BigScience	1760億	可
LLaMA	Meta	70億、130億、330億、650億	不可
LLaMA2	Meta	70億、130億、700億	可
Stanford Alpaca	スタンフォード大学	70億	不可
Falcon	Technology Innovation Institute	70億、400億	可

参考文献4)より

オープンソースのLLMも多数存在

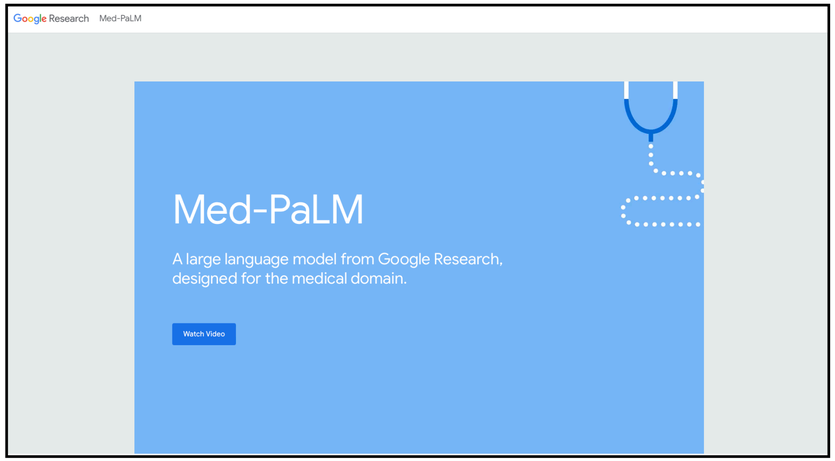
商用利用可能であれば学習済みモデルをファインチューニングして特定のタスクに特化させることが可能

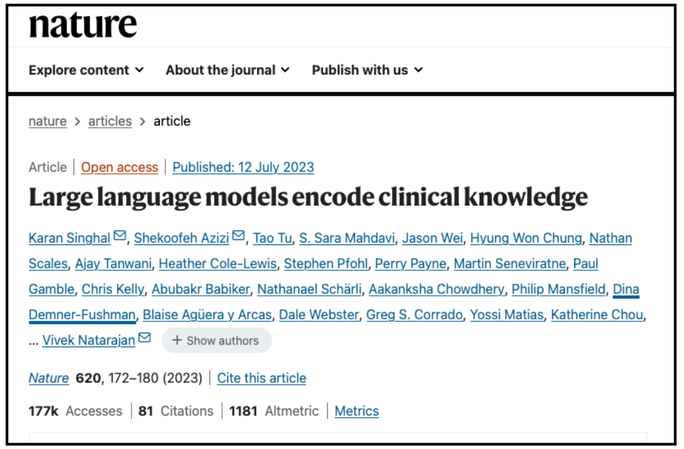
ヘイト表現などの出力調整はされていないことが多いの注意が必要

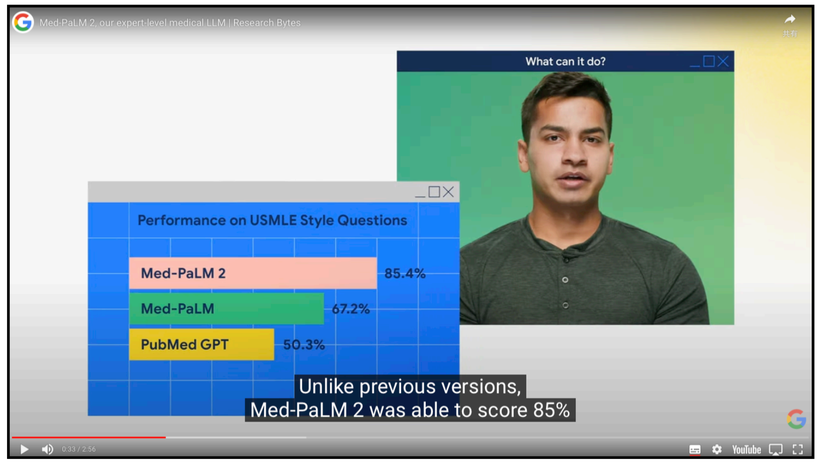
(ex LLamaベースの「Alpaca」は安全性の問題から1週間で公開停止)

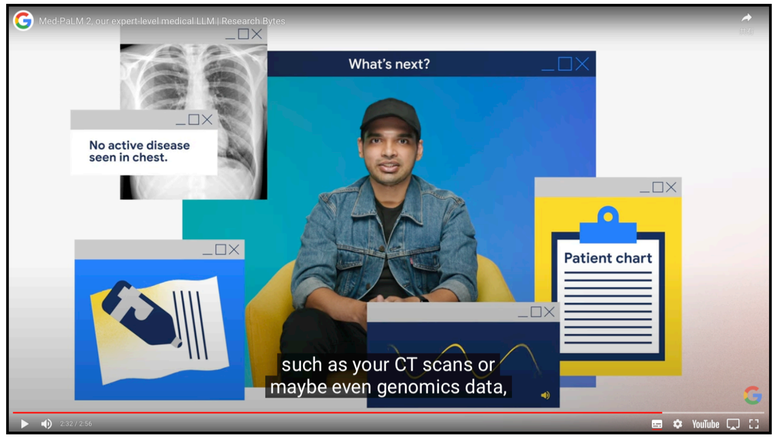
医療とテキスト生成

Googleが医療用LLM「Med-PaLM」を2022に発表
→米国の医師国家試験USMLEで合格ライン(60%)を上回る67.2%を記録
→2023年3月に発表した「Med-PaLM2」では85.4%を記録









今後はマルチモーダルへ

https://sites.research.google/med-palm/

生成AIとその応用

ChatGPTは生成AIの1つのアプリケーションに過ぎない

代表的なサービス

生成AI

テキスト生成AI
- ChatGPT
- Bing AI
- Bard
- Notion AI
- Claude2

画像生成AI
- Stable Diffusion
- DALL-E
- Midjourney

音楽生成AI
- MusicLM
- MugicGen
- WavTool

動画生成AI
- Imagen Video
- Make-A-Video
- Gen-2

UI/Webサイト生成AI
- Galileo AI
- Uizard

参考文献4)より

画像生成AIのモデル

深層学習を使った代表的な生成モデル(深層生成モデル)

- VAE(変分自己符号化器)
- GAN(敵対的生成モデル)
- 自己回帰モデル
- 正規化フロー
- 拡散モデル

GAN(敵対的生成モデル)

無(ノイズ)から画像



Tero Karras, et al. arXiv(2019)

写真が証拠になる時代は終わった?

GAN(敵対的生成モデル)

GAN : Generative Adversarial Network

Generator

作る(偽札)



VS

Discriminator

見抜く(警察)



GAN(敵対的生成モデル)

Generator
生成器

Discriminator
識別器

ノイズ



偽画像

真の画像

偽画像



真の画像か
偽画像か分類

初めはデタラメな画像を作る
→学習して真の画像に近づける

初めは区別出来ない
→学習して偽画像を見分けるようにする

2つのニューラルネットワークが競争的に学習させて画像を生成させる

GAN(敵対的生成モデル)

Generator
生成器

ノイズ



偽画像

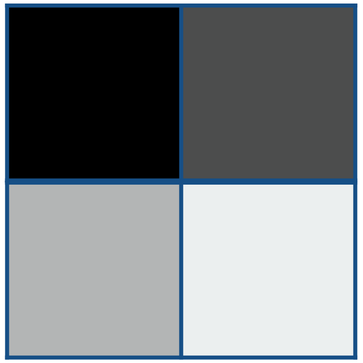
ノイズとは単なる乱数のベクトルのこと

ex)

[1, 10, 100, 200]



[[1, 10]
 [100, 200]]



GANも派生系が沢山存在

GAN → DCGAN → CycleGAN → ProGAN → BigGAN → StyleGAN

拡散モデル

テキストから画像を生成する代表的な生成AI

DaLLE-3、Stable diffusion、Mid journey

画像にノイズを加えてその過程の確率分布を得る(≒パラメタを得る)

今度はノイズからノイズを除去していく過程を学習して鮮明な画像を生成する



Stable diffusionは拡散モデルでコードも公開されている

DaLLEも拡散モデルが使用されている

Mid journeyは不明？

GANと拡散モデル

GAN:

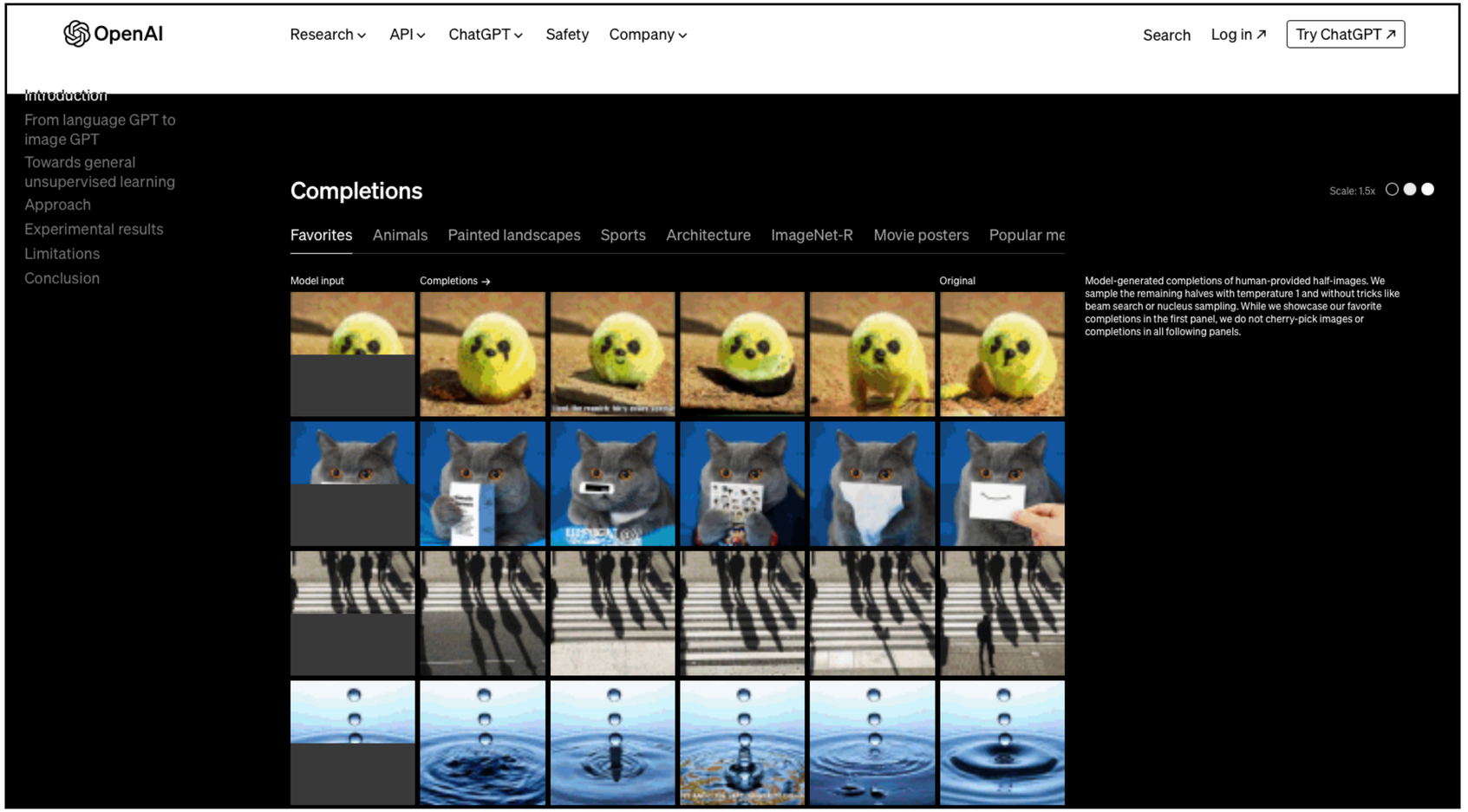
構造: GANは二つのネットワーク、生成器（Generator）と識別器（Discriminator）から成る
動作: 生成器は新しいデータを生成し、識別器はそのデータが本物か生成物かを判断する
二つのネットワークが競争しながら学習する
特徴: GANは特に画像生成において高い品質の結果を生み出すことができるが、訓練が難しく、
モード崩壊（生成されるデータの多様性が失われる現象）が起こりやすい

拡散モデル:

構造: 拡散モデルはデータを段階的にノイズ化し、その後、逆のプロセスで元のデータに戻すことに基づく
動作: 最初にデータにノイズを加え、その後、ノイズを徐々に取り除くことで元のデータを再構築する
特徴: 拡散モデルはより安定した学習プロセスを持ち、多様なデータを生成することができるが、
生成に時間がかかる

その他、画像生成AIの応用例

Image GPT(2020年)

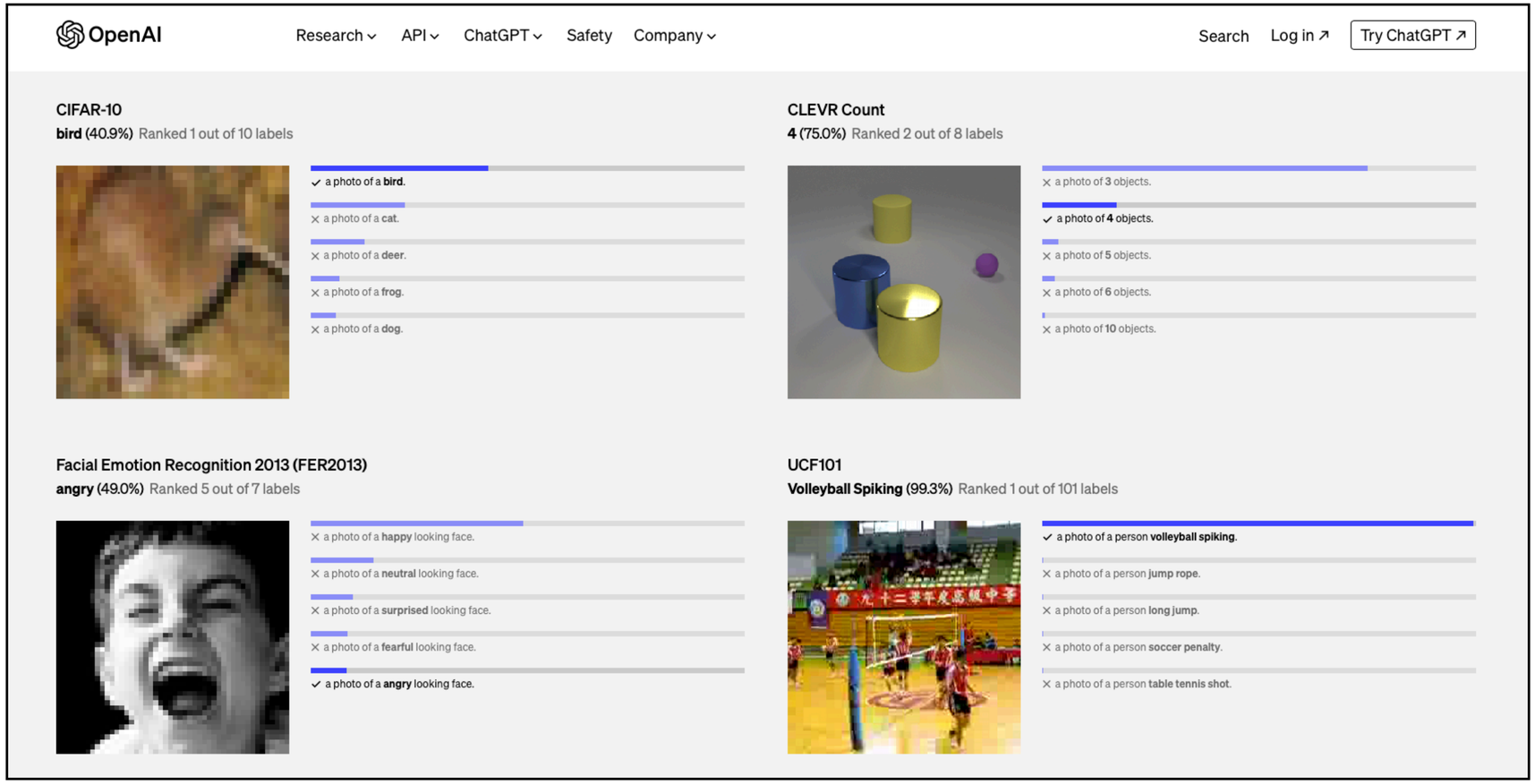


半分の画像から残りの半分の画像を生成

その他、画像生成AIの応用例

CLIP (2021年)

画像とテキストの関連性をランクづけする深層学習

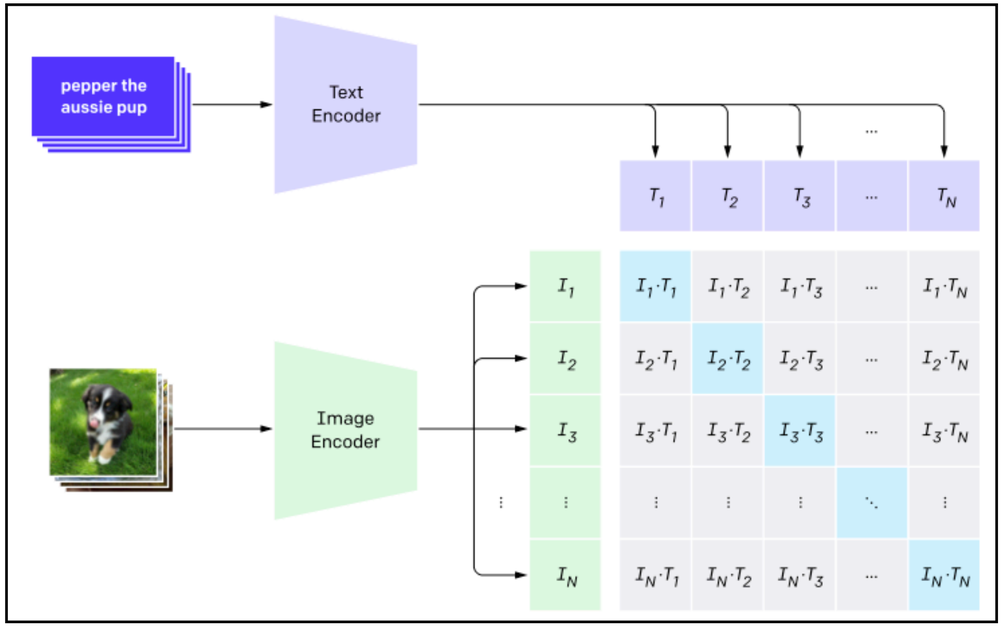


https://openai.com/research/clip

その他、画像生成AIの応用例

CLIP (2021年)

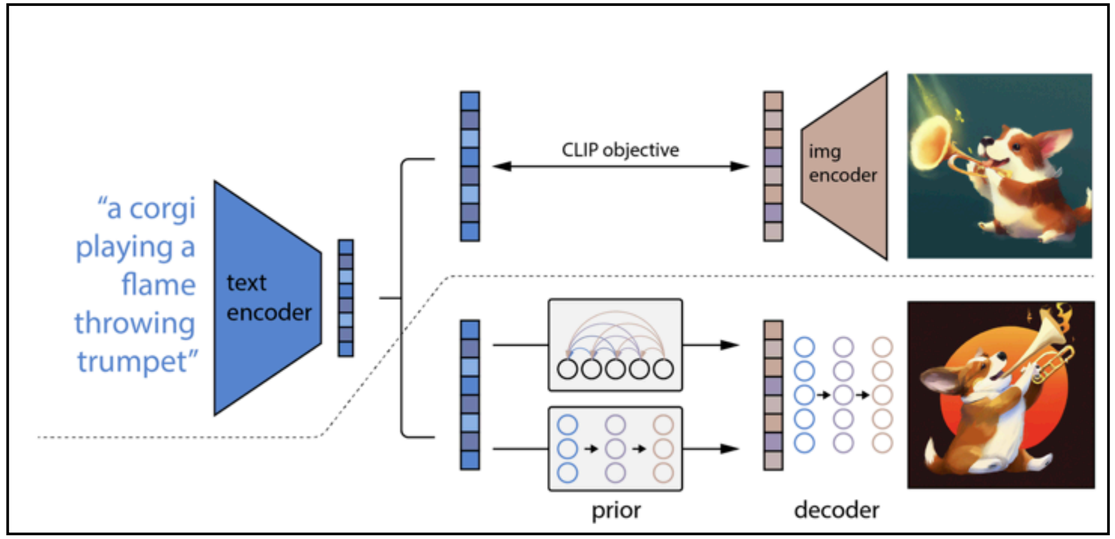
テキストのエンコーダーはtransformer



https://openai.com/research/clip

画像のエンコーダーはResNet版とVit版

DALLE-E2はCLIPと拡散モデル(GLIDE)を採用
ランク付けしたデータを元に画像生成



Aditya Ramesh.et al.arXiv(2022)

画像とテキストの関連性をランクづけする深層学習

自動着色

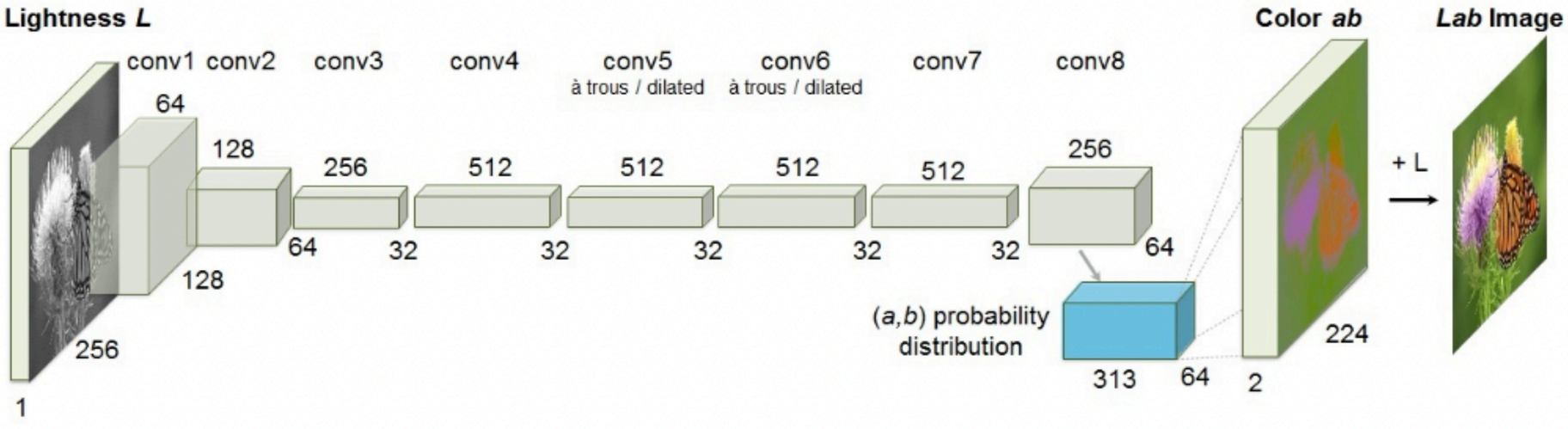


CNNとエンコーダ・デコーダアーキテクチャによる白黒のカラー変換

http://richzhang.github.io/colorization/

自動着色

CNNとエンコーダ・デコーダアーキテクチャによる白黒のカラー変換



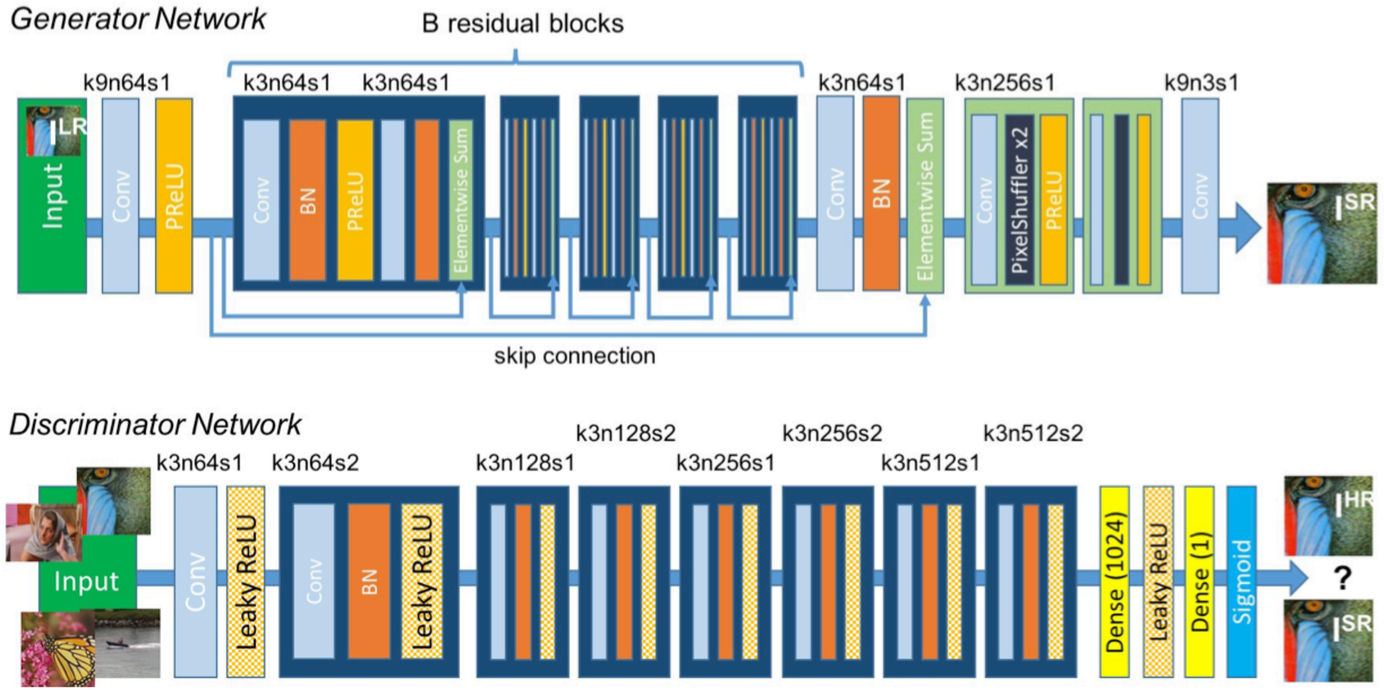
256×256を32×32まで圧縮して224×224まで再構築

RGBをLABという色表現に変換
L(明るさ)を入力してAB(色合い)を予測
最後にABにLを足してLABとして出力

http://richzhang.github.io/colorization/

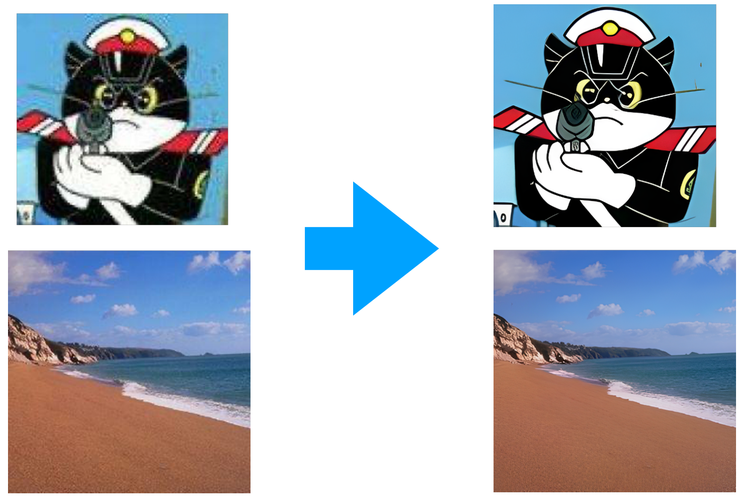
超解像

画像を高画質に変換



GANの派生系であるSRGAN(Super Resolution GAN)

元画像を低解像にする
低解像をモデルが高解像にして元画像との誤差を計算

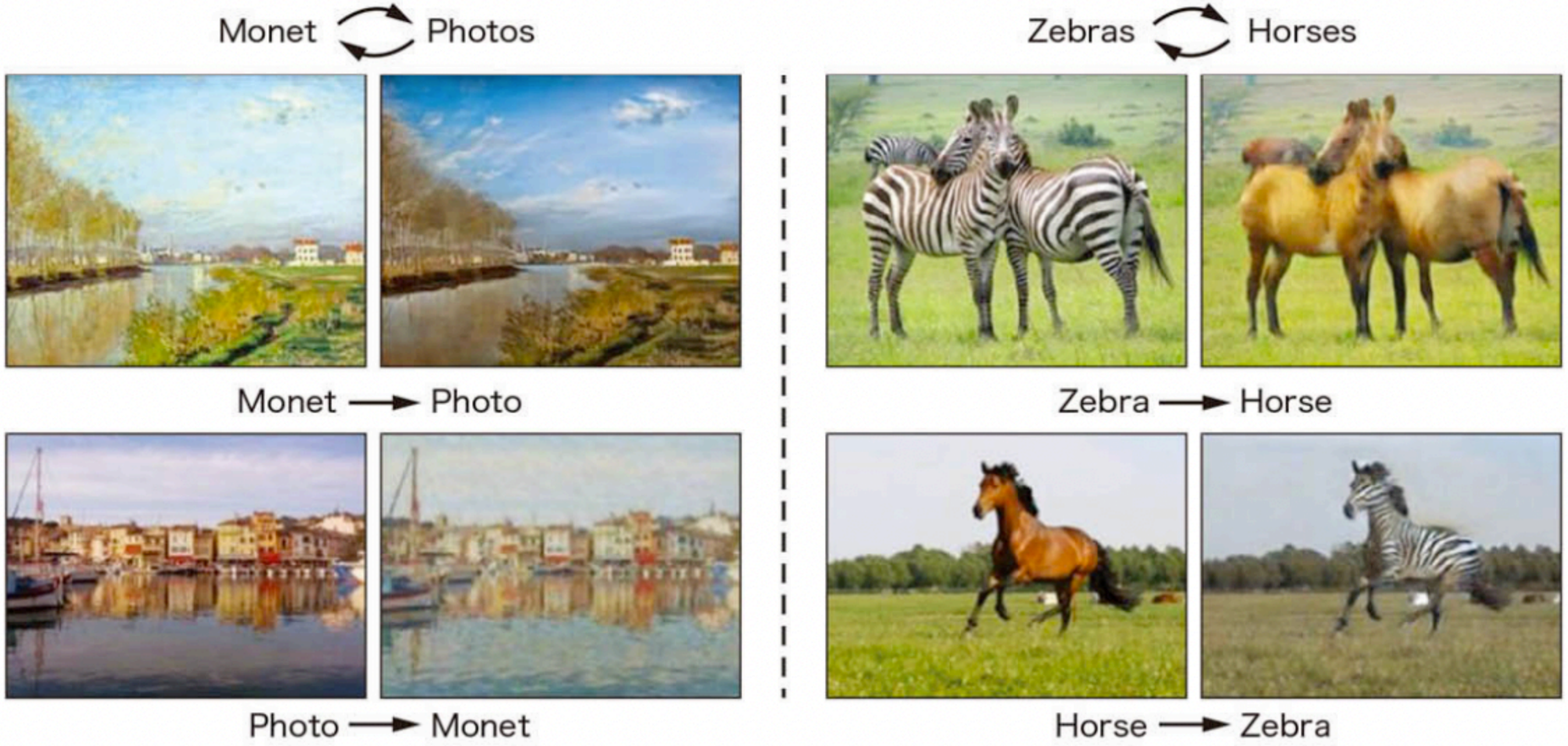


さらに改良版ESRGAN
(Enhanced SRGAN)

https://github.com/xinntao/ESRGAN

その他、画像生成AIの応用例

画像変換 (cycleGAN)



AIの注意点　　倫理的問題　DeepFake



著名人のファイク動画が
SNSで乱用され社会問題に

この動画は簡単に作れることを
示すデモとして公開

作る側と使う側の双方の倫理的配慮が必要

AIの注意点　　ハルシネーションに注意

米国の弁護士がChatGPTで判例をリサーチし、裁判所に提出
→実在しない判例だった
→ChatGPTにその判例が本当に実在するか聞くと、
「実在します。LexisNexisやWestlawなどの信頼できる法律関連のデーターベースで見つけることができます」と回答して鵜呑み

ChatGPTは堂々と嘘をつく

AIの注意点　　ハルシネーションに注意

python　Copy code

```python
temp4 = tokenizer2.batch_encode_plus(text_temp,
                                     max_length=5,
                                     truncation='only_second'
                                    )
temp4
```

この設定により、各テキストの後ろからトークンが切り詰められ、結果は以下のようになります：

lua　Copy code

```lua
{'input_ids': [[2, 879, 465, 14611, 3], [2, 2116, 491, 465, 3]], 'token
```

したがって、`truncation`オプションを使用してどの方向からトークンを切り詰めるかを制御することができます。

You
その結果はtemp3の結果と一緒だよ

ChatGPT
申し訳ありません、誤った情報を提供しました。`truncation='only_second'`を設定しても、`temp4`の結果は`temp3`と同じになります。トークンを後ろから切り詰める代替オプションは提供されていないようです。

ChatGPTは堂々と間違う

生成AIによる失業論

OpenAIとペンシルベニア大学の研究者による論文

「GPT’s are GPTs : An Early Look at the Labor Market Impact Potential of Large Language Models」

- LLMを使用すると、米国の全労働者の約15%の仕事が同じ品質を維持しつつ、著しく早く完了する。さらにLLMを組み込んだソフトウェアやツールを利用すると、その割合は47~56%にまで増加する
- 米国の全労働者の約80%は、自身の仕事のうち少なくとも10%が影響を受ける
- 米国の全労働者の約19%は、自身の仕事のうち少なくとも50%が影響を受ける
- 賃金別では特に高賃金の職種が多くの影響を受ける
- スキル別ではプログラミングやライティングのスキルを必要とする職種ほど影響が大きく、サイエンススキルやクリティカル・シンキングのスキルが重要な職種は影響が少ない

AIによって仕事が完全になくなるのではなく、補完する割合が職種によって異なるという結論

ChatGPT資本主義 / 城田真琴 / 東洋経済新報社

生成AIによる失業論

サム・アルトマン(OpenAI CEO)

「ChatGPTを人間の置き換えではなく、ツールとして見る」ことを推奨

博報堂DYグループ

プロンプトエンジニアリングの育成に投資することを表明

傘下の博報堂テクノロジーズの300人を対象にプロンプトエンジニアリングのカリキュラムを提供

さらに教育プログラムを展開し、グループ全体で1000人のプロンプトエンジニアの養成を発表

伊藤忠商事

「生成AI研究ラボ」を設立し、全社員の生成AIの活用、業務の生産性向上の検証の開始を発表

生成AI前提での働き方やビジネスプロセスの見直しが必要

ChatGPT資本主義 / 城田真琴 / 東洋経済新報社

著作権問題

日本の著作権法

通常(機械学習の目的以外)、著作物を著作権者に無断で利用する(複製、公衆送信、頒布など)と、著作権法違反となる

第30条の4
「著作物は、次に掲げる場合そのほかの当該著作物に表現された思想又は感情を自ら享受し又は他人に享受させることを目的としない場合には、その必要と認められる限度において、いずれの方法によるかを問わず、利用することができる。
ただし、当該著作物の種類及び用途並びに当該利用の様態に照らし著作権者の利益を不当に害することとなる場合は、この限りではない。
(中略)
二　情報解析（多数の著作物その他の大量の情報から、当該情報を構成する言語、音、映像その他の要素に係る情報を抽出し、比較、分類その他の解析を行うことをいう。第四十七条の五第一項第二号において同じ。）の用に供する場合

日本は情報解析が目的なら著作権を有するコンテンツを許諾なく利用できる？

ChatGPT資本主義 / 城田真琴 / 東洋経済新報社

伝えたいこと(私見)

- ChatGPTをはじめとするAIはどんどん進化してより身近な存在になると思います
  ドンドン使用してAIを使いこなせるようになりましょう

- 演習を通して学んだMLP、CNN、Transformerなどは今の多くのAIでも実際に
  使われています。理解を深めることでAIの活用の幅が広がると思います

- ChatGPTなどの対話型AIは必ずしも正解を教えるとは限りません
  僕は自分よりも遥かに知識を持った相談相手だと思ってます

- 入門、応用を通してPythonを使ってきました
  僕のChatGPTの1番使用頻度が高い使い方はプログラミング補助です
  Huggingfaceなど、Pythonが少し出来れば自分のPCで試せるアプリケーション
  は沢山あります(どんどん試してみてください)

長い間お疲れ様でした

参考文献)

1) ディープラーニングを支える技術　　/ 岡野原大輔 / 技術評論社
2) ディープラーニングを支える技術２　/ 岡野原大輔 / 技術評論
3) 生成Deep Learning / David Foster(松田晃一、小沼千絵訳) / O’REILLY
４)ChatGPT資本主義 / 城田真琴 / 東洋経済新報社