# at-5.画像分類

(ディープラーニングのシステムとプログラミング)

(全１２回)

https://www.kkaneko.jp/ai/at/index.html

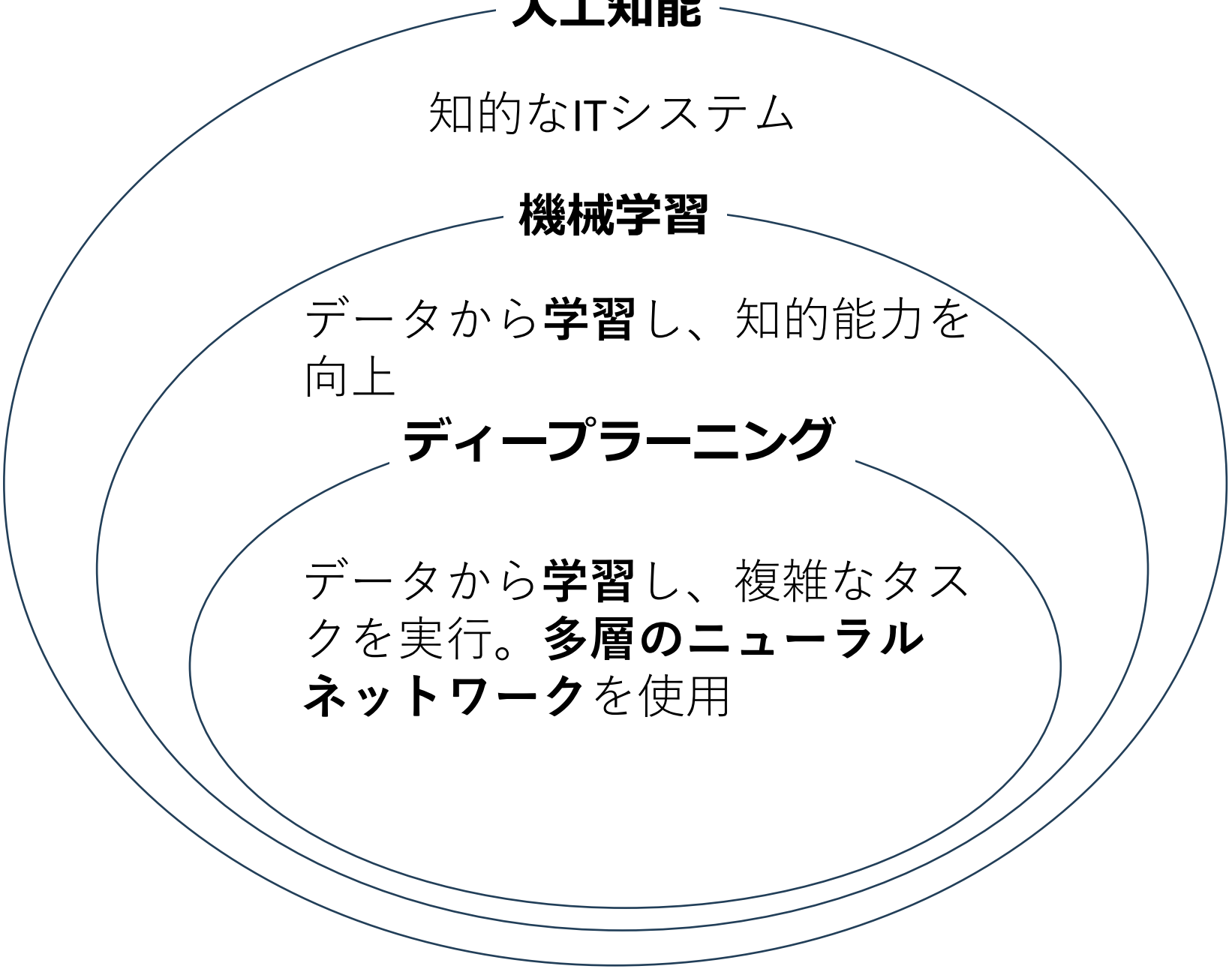
金子邦彦

1. **ディープラーニング**による**画像分類**の**基礎**から**最新技術まで**を解説
2. **畳み込みニューラルネットワーク(CNN)**によって、AI による画像理解が飛躍的に進歩
3. ResNetなどの技術により、より**深いニューラルネットワーク（ディープニューラルネットワーク）**が実現可能に
4. **学習済みモデル**の**転移学習**や**ファインチューニング**で、様々なタスクに適用

# アウトライン

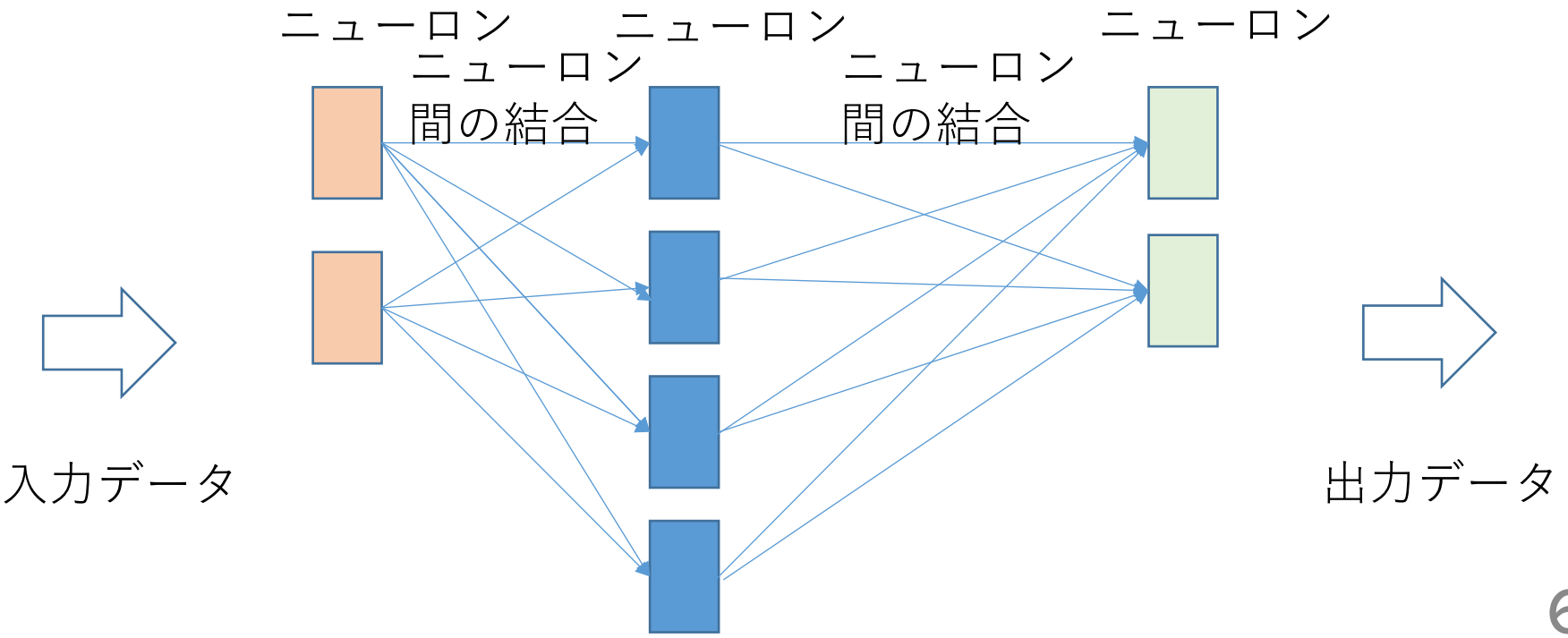
1. イントロダクション
2. コンピュータによる画像理解
3. 画像データの扱い
4. 畳み込みニューラルネットワーク(CNN)の基礎
5. 畳み込み
6. 全結合層
7. 畳み込み層
8. プーリング層
9. CNN Explainer のデモ

# 5-1. イントロダクション


人工知能
知的なITシステム
機械学習
データから学習し、知的能力を向上
ディープラーニング
データから学習し、複雑なタスクを実行。多層のニューラルネットワークを使用

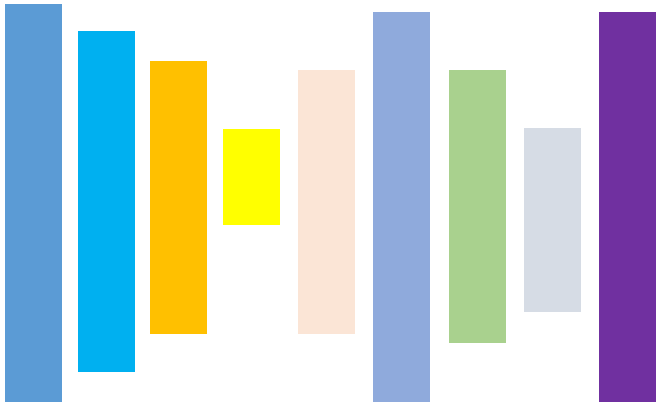
# ニューロンとニューラルネットワーク

- **ニューロン**は、**ニューラルネットワーク**の基本的な構成要素
- 一つ一つの**ニューロン**は、データの受け取り、処理、伝達を行う
- **ニューラルネットワーク**は、これらの**ニューロン**が多数組み合わさったもの



**ディープラーニング**に「**ディープ**」とついているのは、**多層のニューラルネットワーク**を使用するため

層の数が少ない



層の数が多い（ディープ）

# ディープラーニングまとめ


- **ディープラーニング**は**機械学習**の一種であり、人工**ニューラルネットワーク**を使用して**データから学習**し、複雑な**タスクを実行**する技術

- 「ディープ」の名前は、**多層のニューラルネットワーク**を使用することに由来

- ディープラーニングが広く利用される理由は、**多様なデータに適用**でき、**さまざまなタスク**で高性能を発揮するため。
  例：**画像認識**、**自然言語処理**、**音声認識**など。

# 5-2. コンピュータによる画像理解

# コンピュータによる画像理解

コンピュータが画像を理解する

# コンピュータによる画像理解

- **一般的な画像が対象となる**

  （実験室で撮影などの制約が無い）
- **さまざまな応用**：スマホ，デジカメ，自動車，ロボット
- **さまざまな種類**：画像分類，物体検出，セグメンテーション，顔画像処理，姿勢推定など

# 画像理解の応用例：医療分野

## 画像内の差異の抽出（傷，汚れ，病変など）

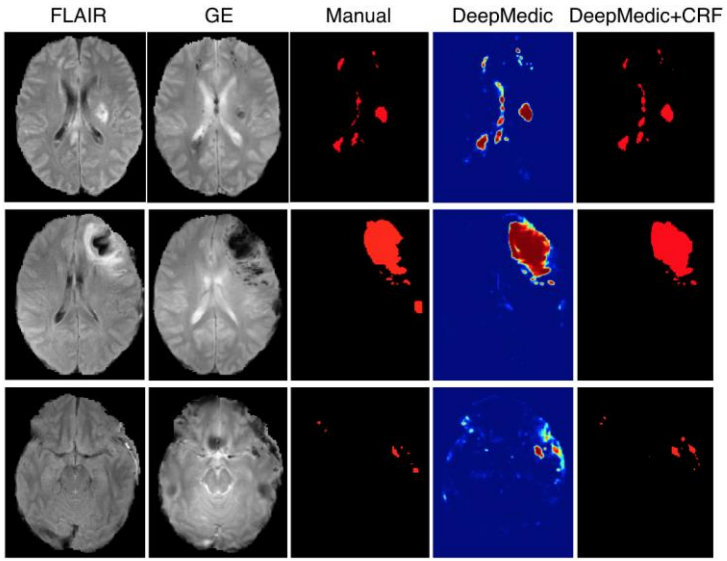



Figure 11: Three examples from the application of our system on the TBI database. It is capable of precise segmentation of both small and large lesions. Second row depicts one of the common mistakes observed. A contusion near the edge of the brain is under-segmented, possibly mistaken for background. Bottom row shows one of the worst cases, representative of the challenges in segmenting TBI. Post-surgical sub-dural debris is mistakenly captured by the brain mask. The network partly segments the abnormality, which is not a celebral lesion of interest.

脳内の病変の抽出

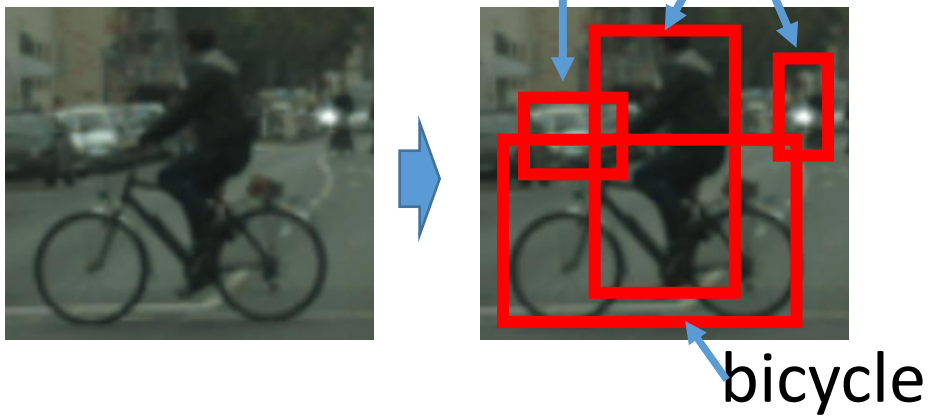
Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation, Konstantinos Kamnitsas, Christian Ledig, Virginia F.J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Daniel Rueckert, Ben Glocker, arXiv: 1603.05959, 2016.

# ① 画像分類

```
Score 0.9827020168304443, Label lab_coat
Score 0.003087261691689491 3, Label syringe
Score 0.002431107917800545 7, Label beaker
Score 0.001660922775045037 3, Label stethoscope
Score 0.000379508855985477 57, Label plate
```

**画像分類の結果は，ラベル**と**確率**
※ 5 つの候補 (top 5) が表示されている

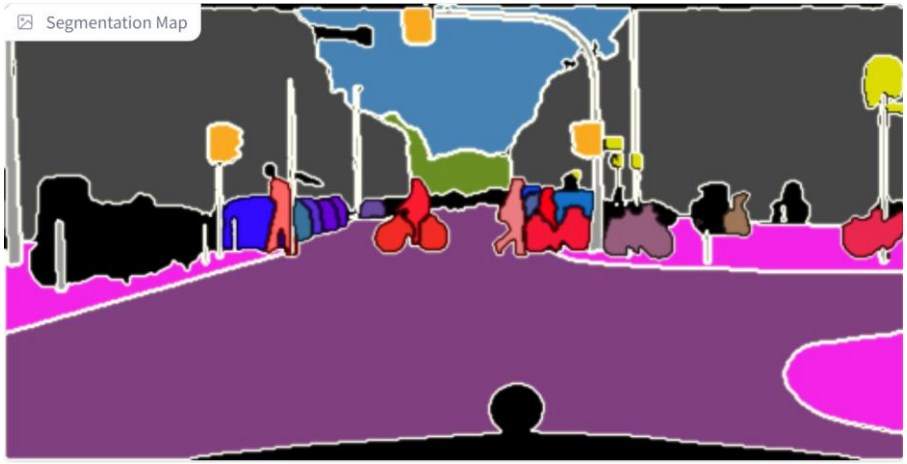
# ② 物体検出




**バウンディングボックス，**
**ラベルを得る**

**バウンディングボックス**は，
物体を囲む最小のボックス（四角形）

# ③ セグメンテーション




物体の形を画素単位で抜き出し


ラベルを得ることもできる

# 画像理解の主な種類

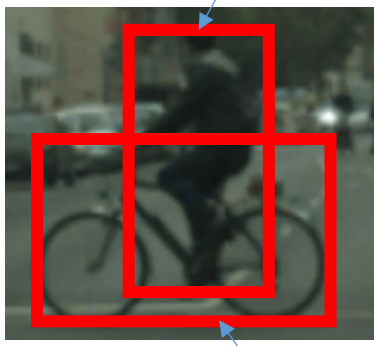
① **画像分類**

「何があるか」を理解

person
bicycle

② **物体検出**

場所と大きさも理解




③ **セグメンテーション**

画素単位で理解

# 画像分類の精度の向上


**ImageNet データセットの画像分類の結果**

- **ディープラーニング**の進展
- **画像分類は，場合によっては，AIが人間と同等の精度**　とも考えられるように

画像分類の誤り率 (top 5 error)

**人間: 5.1 %**

**PReLU による画像分類: 4.9 %**

**（2015年発表）**

文献: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification
arXiv:1502.01852, 2015.

# ここまでのまとめ

**画像理解の主な種類**

- **画像分類**

「何があるか」を理解。結果は「ラベル」として識別

各ラベルに対する「確率」も提供

- **物体検出**

物体の種類、場所、大きさを理解。場所と大きさについての結果は、物体を囲むバウンディングボックス。

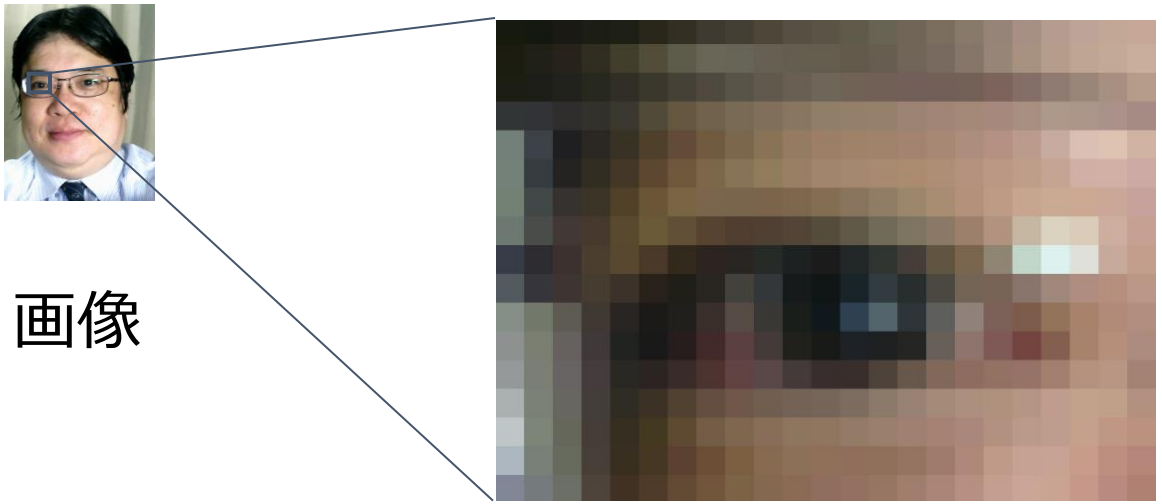
**セグメンテーション**

画素単位で理解。

物体の「形」を詳細に抽出。

# 5-3. 画像データの扱い

# 画像と画素



画像

それぞれの格子が画素

# 画像の種類

カラー画像

**輝度**と**色**の情報

濃淡画像

**輝度**のみの情報

# 濃淡画像でのコード化

画像の輝度の情報

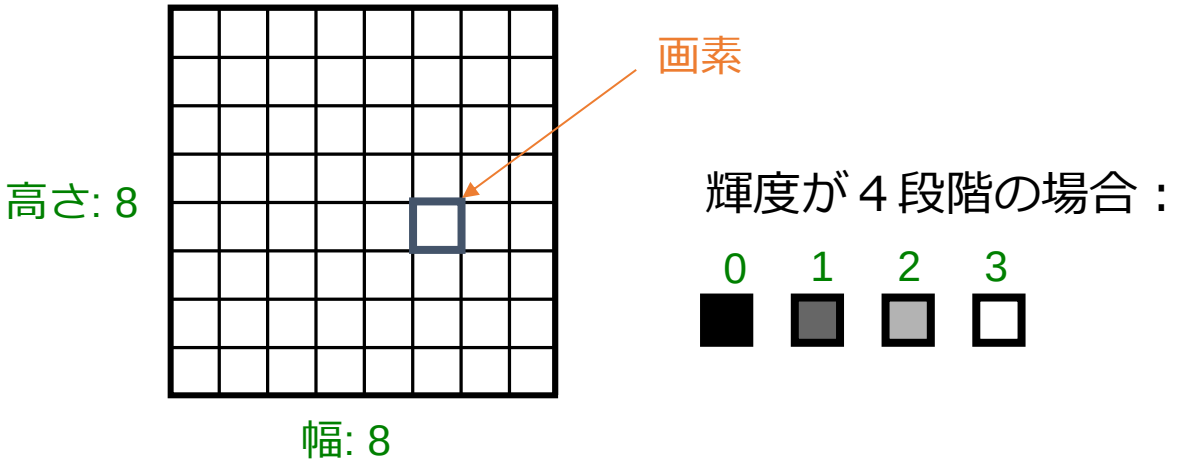
例えば：　黒　＝　0，

暗い灰色　＝　1，

明るい灰色　＝　2，

白　＝　3

のように**コード化**



# R（赤）成分，G（緑），B（青）成分で考える場合

R（赤）成分　G（緑）成分　B（青）成分

画素ごとに
１つの数値

画素ごとに
１つの数値

画素ごとに
１つの数値

すべてあわせて，画素ごとに３つの数値

# 演習 1

**カラー画像の画素**

**【トピックス】**

**カラー画像**

**画素**

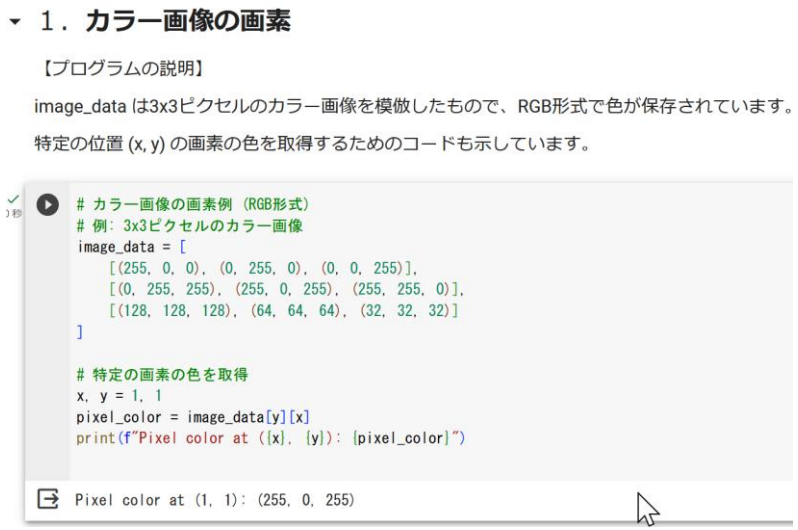
**配列**

# ① Google Colaboratory のページを開く

https://colab.research.google.com/drive/1MMhlrh08-0Byq-U1JlTdDVwMe6dyUcaq?usp=drive_link

② **次**について、プログラムや説明や実行結果が掲載されていることを確認。各自でよく読む。

## １．カラー画像の画素



自習

目的：配列になっている画像データについて、特定の位置の画素の色を取得する方法を学び、理解を深めること。

**指示：異なる (x, y) の位置の画素を取得して、期待通りであるか確認**してみよう

解答例：

x, y = 2, 2

のときは次のように表示される

Pixel color at (2, 2): (32, 32, 32)

```
# 特定の画素の色を取得
x, y = 2, 2
pixel_color = image_data[y][x]
print(f"Pixel color at ({x}, {y}): {pixel_color}")
```

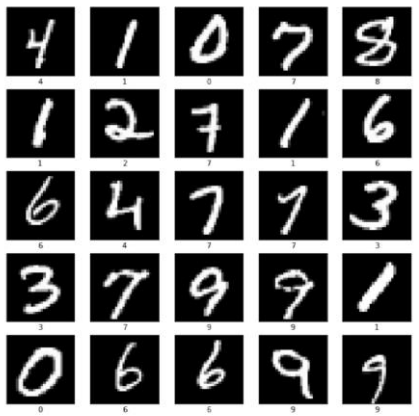
```
Pixel color at (2, 2): (32, 32, 32)
```

# 5-4. 画像分類と畳み込みニューラルネットワーク

# 機械学習と訓練データとプログラム

## 機械学習のプログラム

**学習に使用する訓練データ**
（抜粋）



4 1 0 7 8
1 2 7 1 6
6 4 7 7 3
3 7 9 9 1
0 6 6 9 9

画像 60000枚
（うち一部）

正解 60000個
（うち一部）

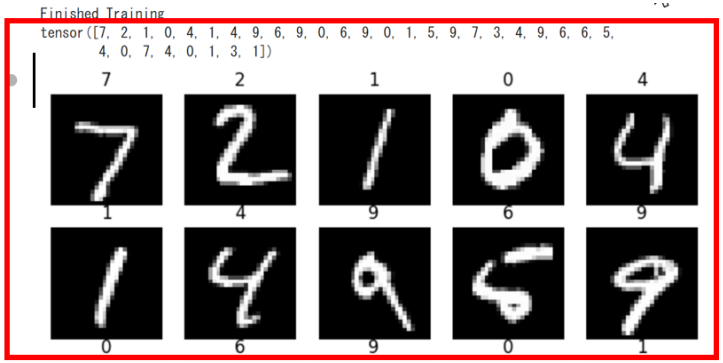
プログラム

データを用いて学習を行う
学習ののち，画像分類を行う

```
[4] !pip install -U scikit-learn matplotlib

    import torch
    import torch.nn as nn
    import torch.optim as optim
    from sklearn import datasets
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    import matplotlib.pyplot as plt

    # データの取得と前処理
    iris = datasets.load_iris()
    X = iris.data
    y = iris.target

    # データの標準化
    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    # 訓練データとテストデータの分割
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    X_train = torch.tensor(X_train, dtype=torch.float32)
    y_train = torch.tensor(y_train, dtype=torch.long)
    X_test = torch.tensor(X_test, dtype=torch.float32)
    y_test = torch.tensor(y_test, dtype=torch.long)

    # ニューラルネットワークの定義
    class Net(nn.Module):
        def __init__(self):
            super(Net, self).__init__()
            self.fc1 = nn.Linear(4, 10)  # 入力は4次元（Irisの特徴量）
            self.fc2 = nn.Linear(10, 3)  # 出力は3クラス

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

    net = Net()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr=0.01)
```

学習の結果、文字認識の能力を獲得

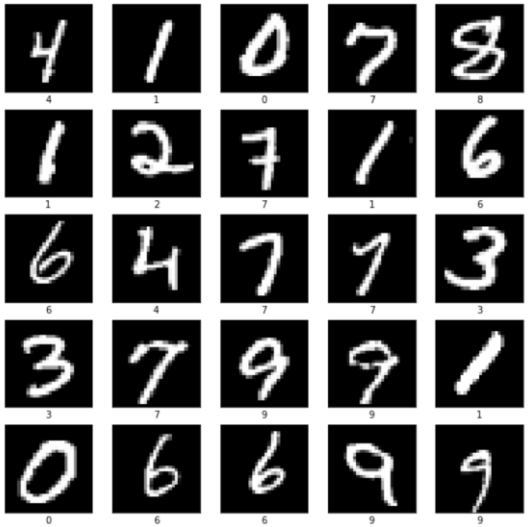
```
Finished Training
tensor([7, 2, 1, 0, 4, 1, 4, 9, 6, 9, 0, 6, 9, 0, 1, 5, 9, 7, 3, 4, 9, 6, 6, 5,
        4, 0, 7, 4, 0, 1, 3, 1])
```



# 手書き文字の MNIST データセット

- **0 ～ 9 の手書き文字．濃淡画像 28 × 28**
- **訓練データ**　（学習用）

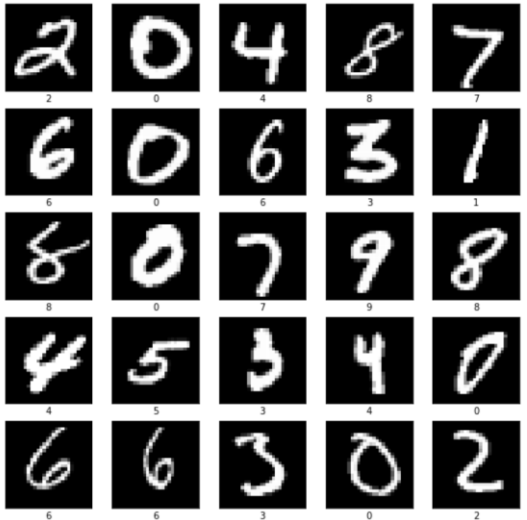
**60000**枚の画像と正解



抜粋

- **検証データ**　（検証用）

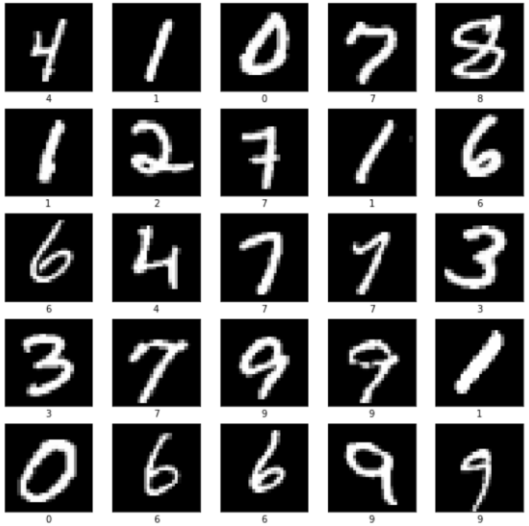
**10000**枚の画像と正解



抜粋

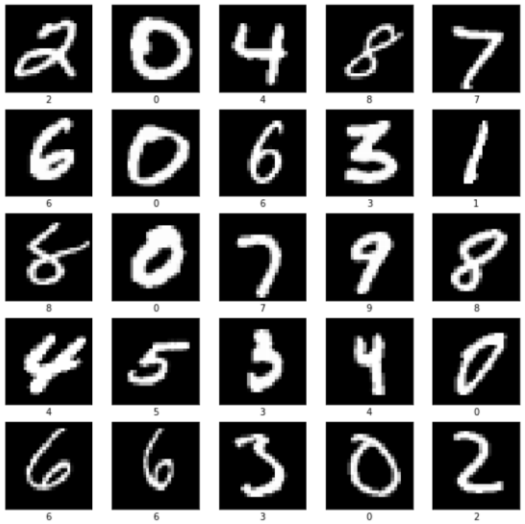
# 手書き文字の MNIST データセットの分類

- 手書き文字の画像を，0 ～ 9 に分類
- **訓練データ**を使用して、**モデルの学習**を行う
- **検証データ**を使用して、**学習されたモデルの検証**を行う



**60000**枚の画像と正解

**訓練データ**



**10000**枚の画像と正解

**検証データ**

# ディープラーニングの利点と弱点

- **層を深くすると複雑な特徴をとらえることが可能**

- 層の深さが増えると「**勾配消失問題**」が発生

勾配消失は、**ニューラルネットワークの入力に近い層（初めの方の層）が正しく学習しづらくなる**問題。

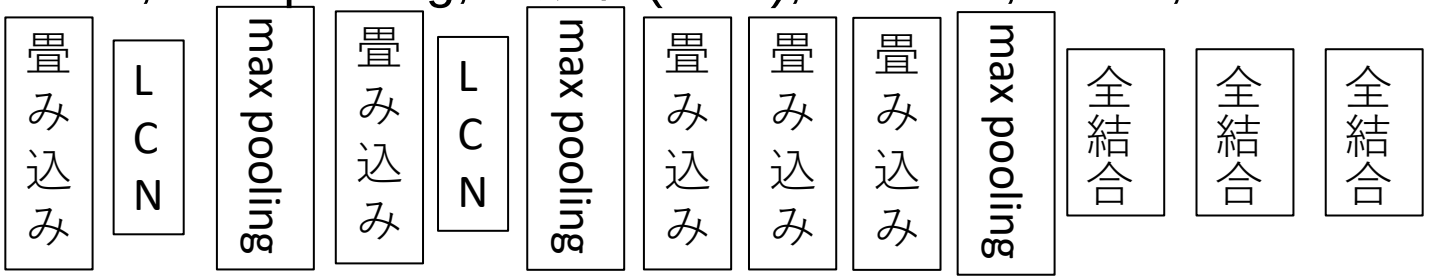
画像理解においても、勾配消失の解決は重要。

（次ページ以降、歴史と、代表的な解決策である ResNet を説明）

# ディープラーニングによる画像分類 AlexNet (2012年)

- **画像分類，教師有り学習，ディープラーニング**
- **特徴：CNN（畳み込みニューラルネットワーク）の導入**

畳み込み, max pooling, 正規化(LCN), softmax, ReLU, ドロップアウト




**訓練データ: 画像約 100万枚以上**（ImageNet データセット，22000種類に分類済み），ILSVRCコンペティション: 画像を **1000 種類**に**分類**

文献: ImageNet classification with deep convolutional neural networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, NIPS'12, 2012.

# ディープラーニングによる画像分類の進展 ①

## CNN（畳み込みニューラルネットワーク）

- AlexNet（2012年）

  **CNN（畳み込みニューラルネットワーク）の導入**

- VGG-16，VGG-19（2014年）

  **プーリングカーネルのサイズ縮小．サイズ縮小の結果，従来より深い CNN を可能に**

- **ResNet (2015 年)**

  **残差結合 (Residual Connection)，Bottleneck Residual Block の導入．30層以上の深い CNN を可能に．** ResNet34, ResNet50, ResNet101, ResNet 152 などの種類

- Xception（2016年）

  **ResNet の畳み込み層を Depthwise Separable Convolution に置き換え**

- EfficientNet（2019年）

  **CNN の深さとチャンネル数と解像度の配分を探索**

（私見）CNNの深さ（層の数）を増やすという方向では完成の域にある．
いまは，チャンネル数，解像度も含む総合的な分析が行われている

# ディープラーニングによる画像分類の進展 ②

## Transformer

- Transformer（2017年）

  **自然言語処理のために Transformer が考案された．Attention を特色とする．**

- Vision Transformer (ViT)（2020年）

  **Transformer を画像理解に使用．CNNと違うもので，畳み込み演算を用いない**

- Swin Transformer （2021年）

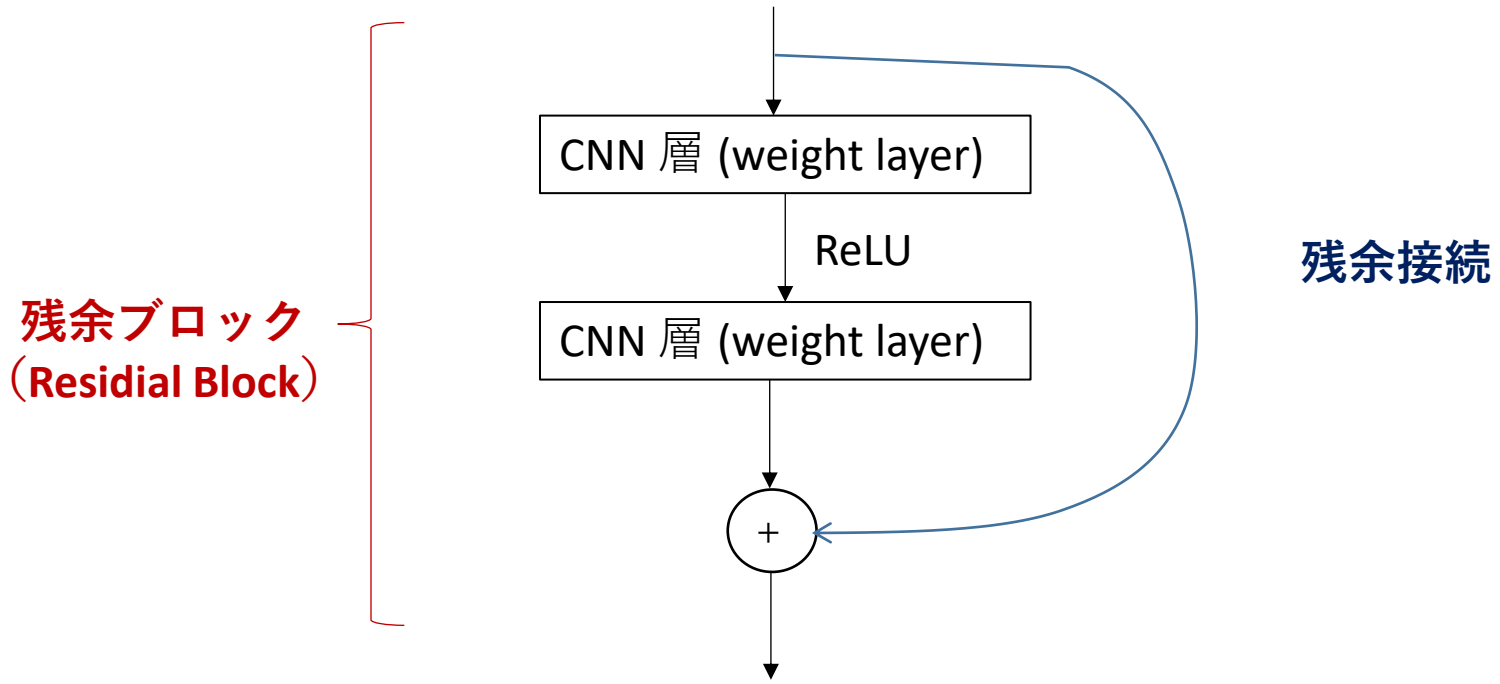
  **vision Transformer に Sifted Windows を導入**

- DiNAT（2022年）

  **vision Transormer で用いられる NA (Neighborhood Attention) の改良.**

# ResNet (2015 年) の基本アイデア

- **残余接続**を導入.
- 層をスキップして，後続の層へ直接データを送る経路を追加. **残余ブロック（Residual Block）** を構成
- **勾配消失問題の緩和**




Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,

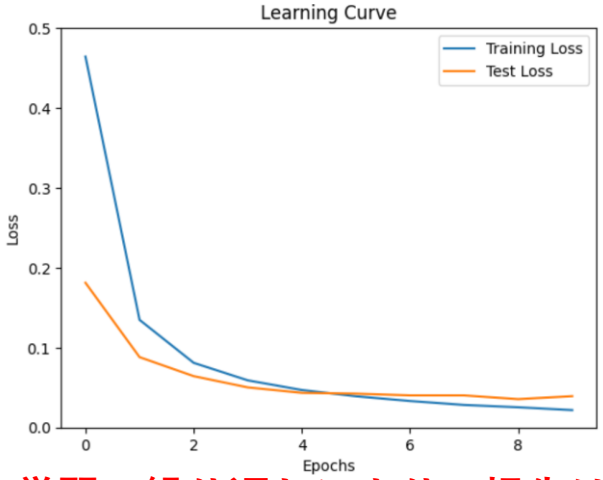
Deep Residual Learning for Image Recognition, IEEE Conference on Computer Vision and Pattern Recognition, 2016.

# ResNet の残余ブロックの効果

畳み込みニューラルネットワークの例

畳み込み層 ⇨ 畳み込み層 ⇨ プーリング層 ⇨ 全結合層 ⇨ 全結合層

手書き文字 (0から9) MNIST の分類での損失

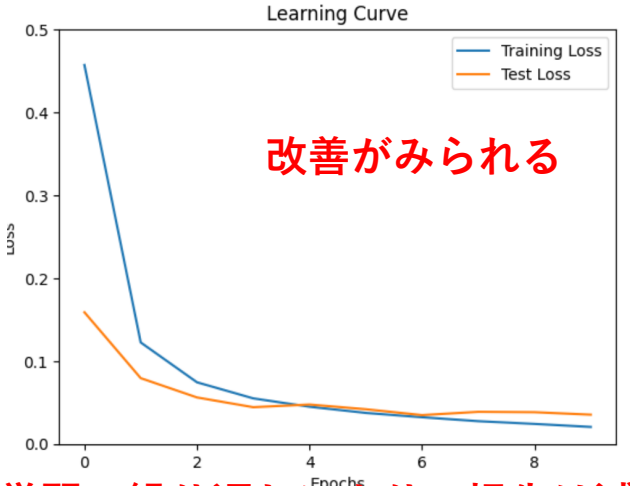



学習の繰り返しにより、損失は減少

ResNet の残余ブロックの導入

**残余ブロック** ⇨ **残余ブロック** ⇨ プーリング層 ⇨ 全結合層 ⇨ 全結合層

手書き文字 (0から9) MNIST の分類での損失




学習の繰り返しにより、損失は減少

# 演習 2

**手書き数字の認識**

**【トピックス】**

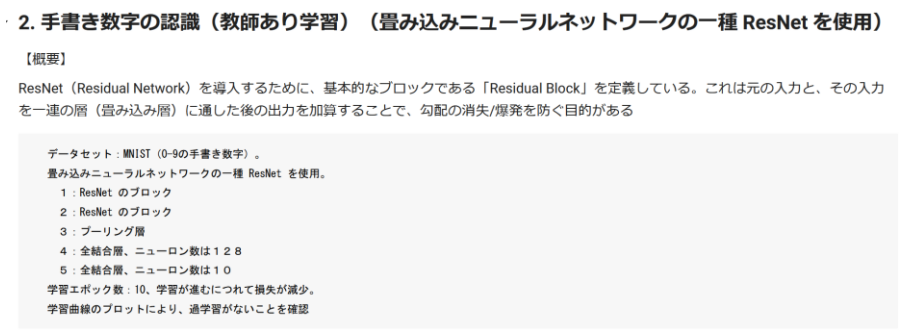
- **畳み込みニューラルネットワークの進展、最新技術**
- **残余ブロック、その効果**

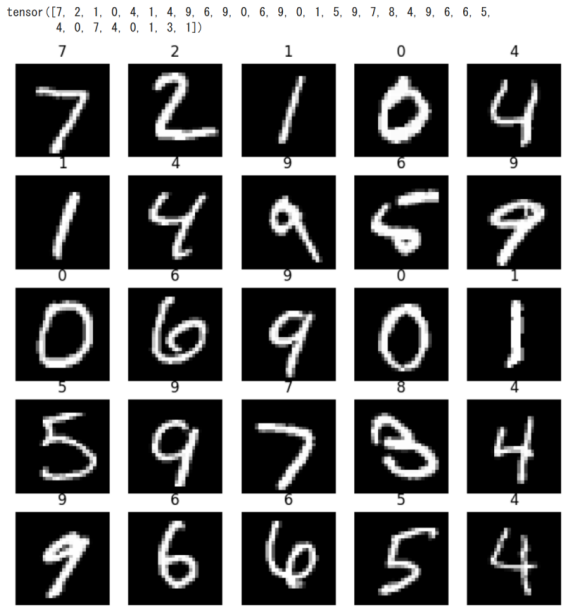
① Google Colaboratory のページを開く
https://colab.research.google.com/drive/18IPPkY96Oc6jkYD2su4cFgWcoYAskLo_?usp=sharing

② プログラムや説明や実行結果が掲載されていることを確認。

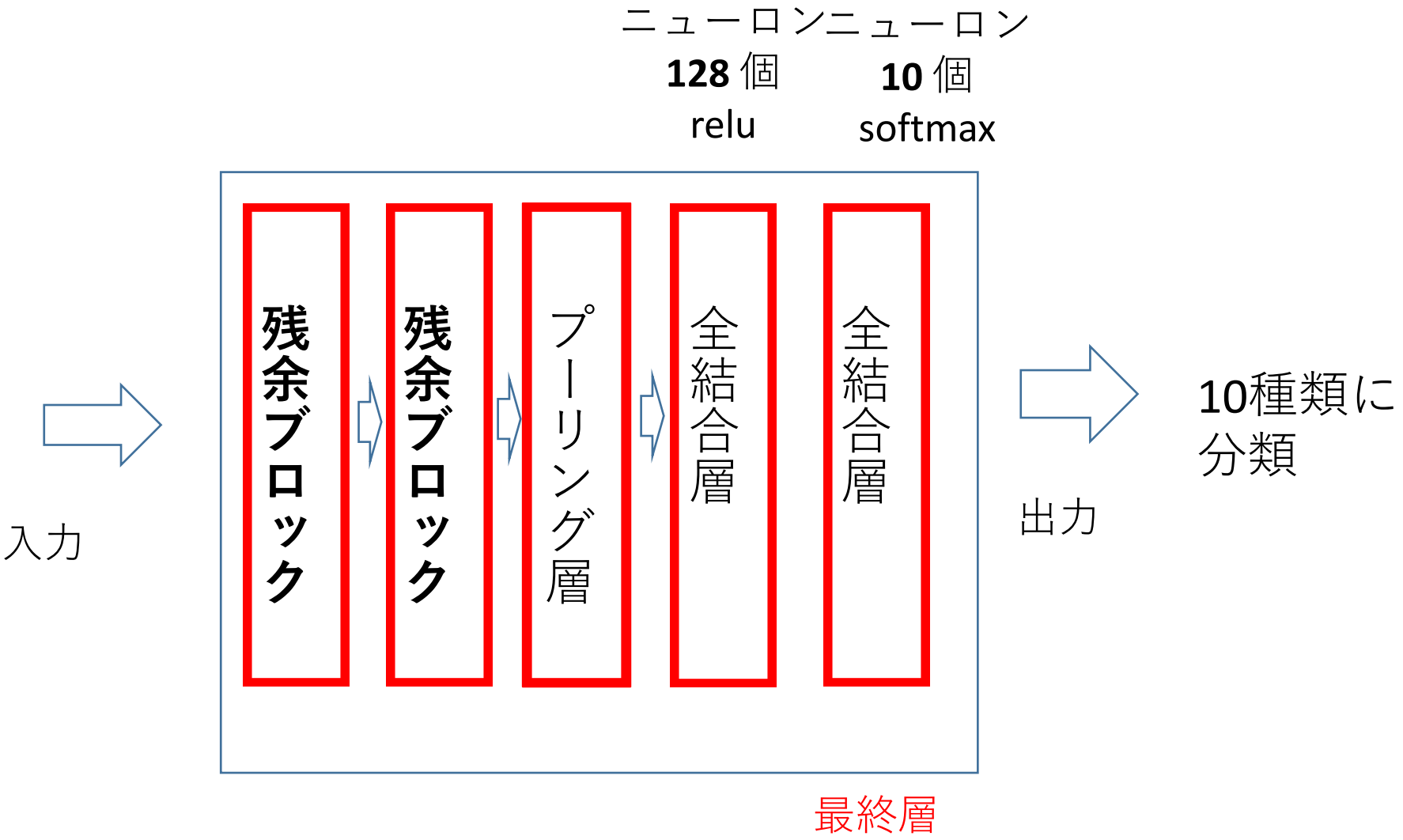
各自でよく読む。

③「**２．手書き数字の認識（教師あり学習）（畳み込みニューラルネットワークの一種 ResNet を使用）**」を確認。**このさき３ページ分で詳細説明**。各自で確認



# 演習２の畳み込みニューラルネットワーク

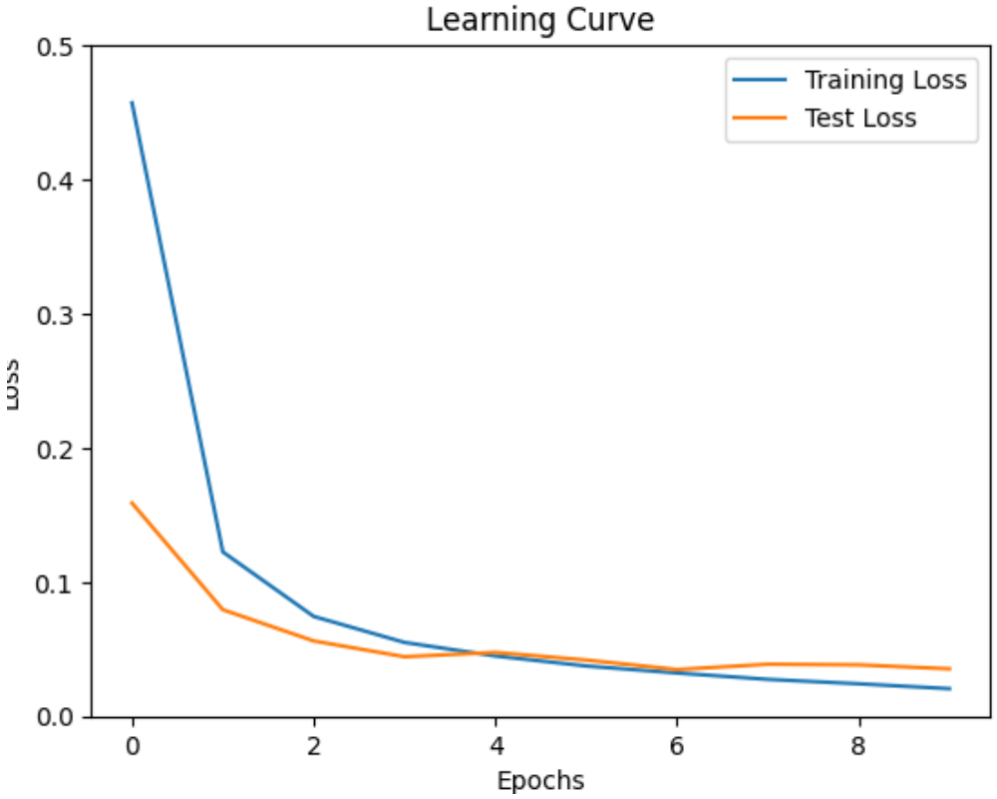


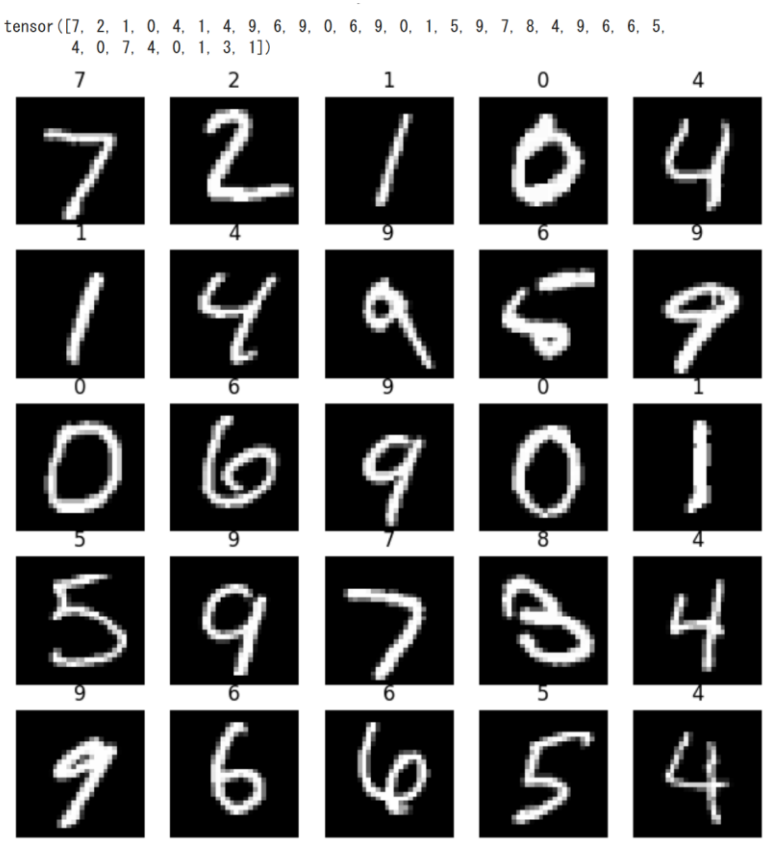
# 演習２で、ニューラルネットワーク作成を行っているプログラムの部分

```
def __init__(self):
super(ResNet, self).__init__()
self.block1 = ResidualBlock(1, 32)  # 残余ブロック
self.block2 = ResidualBlock(32, 64)  # 残余ブロック
self.pool = nn.MaxPool2d(2, 2)  # プーリング層
self.fc1 = nn.Linear(64 * 12 * 12, 128)  # 全結合層、128ニューロン
self.fc2 = nn.Linear(128, 10)  # 全結合層、10ニューロン（クラス数）
```

# 演習２のプログラムの実行結果

**同じ訓練データを用いた学習を１０回繰り返し．**
**そのとき，検証データで検証**




**学習**の**繰り返し**ごとに，訓練データや検証データでの**損失**の変化を確認

# ここまでのまとめ

**手書き文字 MNIST データセット**

- 0～9の手書き数字、28×28ピクセル、60,000枚の訓練データ、10,000枚の検証データ

**手書き文字 MNIST データセットの分類**

- 数字0～9に分類、訓練データでモデルを学習、検証データでモデルを検証

**ディープラーニングの利点と弱点**

- 深い層で複雑な特徴を捉えるが、**勾配消失**の問題が発生することがある
- **勾配消失**により学習が困難になる

**ResNet (2015年)**

- 残余接続と残余ブロックを導入
- 30層以上の深いCNNを実現
- 勾配消失の緩和

**ResNet の残余ブロック**

- 層をスキップして，後続の層へ直接データを送る経路により、勾配消失を緩和

# 6-3. 学習済みモデルの活用

# ディープラーニングの学習済みモデル

- 有名なデータで**学習済みのモデル**は，**インターネットで公開**されていることが多い

# ディープラーニングの学習済みモデル

- 大規模データセットで**学習されたモデル**
- ニューラルネットワークの**パラメータ**（重み、バイアスなど）は**最適化済み**
- **特定のタスク**（所定の 1000種類の画像分類など）に**特化**して**最適化**されている
- **転移学習**や**ファインチューニング**を利用して、**異なるタスクにも適用可能**
- **転移学習**や**ファインチューニング**にはそのためのデータが必要。**少量のデータ**でも**効果的に動作する**場合がある

# ディープラーニングの学習済みモデルの活用


- **コスト削減**

学習済みモデルは、**大規模データセット**で**学習済み**のため、
最初から学習するよりも時間や手間や設備が省ける

- **高い性能**

**大規模データセットでの学習**により、**優れた性能**を持つ

- **転移学習、ファインチューニング**

学習済みモデルの層を再利用し、異なるタスクにも適用可能

# 学習済みモデルの活用シーン


- **迅速な技術検証**

新しい技術を素早く試したい

- **学習済みモデルの継続利用**

既存の学習済みモデルが自分のタスクに適している場合

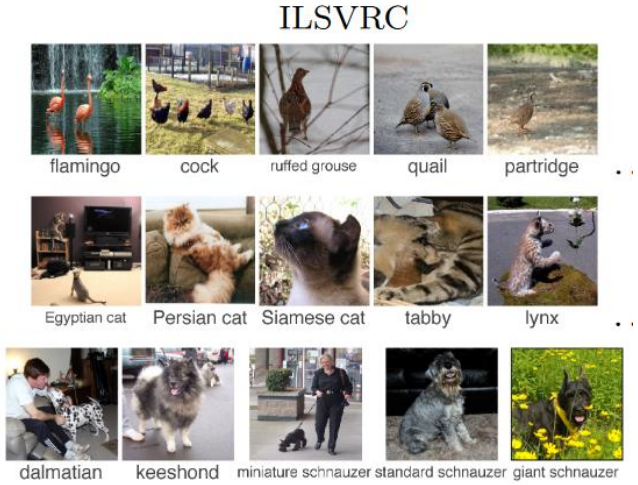
例：顔検出、姿勢推定

- **転移学習、ファインチューニングの導入**

既存の学習済みモデルを新しいタスクに適用させるための転移学習、ファインチューニングを予定している

# ImageNet-1K データセット

- **画像の総数**：カラー画像 **約120万枚**
- **クラス数**：**1000 種類**に分類済み
- **ラベル付きの画像：**画像が、ある特定のクラスのオブジェクトを含んでいるか（有無）のデータ
- 画像のうち 45万枚については、画像内のオブジェクトの位置と大きさのデータもある




```
Class ID: 0, Class Name: tench
Class ID: 1, Class Name: goldfish
Class ID: 2, Class Name: great white shark
Class ID: 3, Class Name: tiger shark
Class ID: 4, Class Name: hammerhead
Class ID: 5, Class Name: electric ray
Class ID: 6, Class Name: stingray
Class ID: 7, Class Name: cock
Class ID: 8, Class Name: hen
Class ID: 9, Class Name: ostrich
```

1000種類のうち 0〜9番のクラス名

文献: ImageNet Large Scale Visual Recognition Challenge
Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei

# ImageNet-1K で学習済みのモデルの活用

**１．迅速な技術検証**

画像分類の新しい技術を素早く試すために、学習済みモデルを利用。

**２．学習済みモデルの継続利用**

ImageNet-1k の 1,000種類のクラスが、自分の目的とする画像分類タスクと一致する場合、そのまま使用

**３．転移学習・ファインチューニングの導入**

ImageNet-1kのクラスが目的と一致しない場合でも、転移学習やファインチューニングの実行が可能

# 授業の学ぶ意義と満足感

- **最先端のディープラーニング**による**画像分類**技術を学び、情報工学への理解を深められる
- **画像データの扱い**、**CNN**など、実践的な知識が身につく
- **勾配消失問題などの課題**とその**解決策**を学ぶことで、**ディープラーニングの進展**を理解できる
- **学習済みモデルの活用**を知る。将来の各自の応用につながる
- **学ぶ楽しさ**を感じながら、知識とスキルを向上させ、学びの達成感が更なる学習意欲につながる