3-4　深層学習の基礎と展望

東京大学 数理・情報教育研究センター
2021年5月7日

概要

- 深層学習の基礎を扱う．深層学習で用いられるニューラルネットワークモデルの原理と学習方法を解説したのち，応用事例の紹介をする．

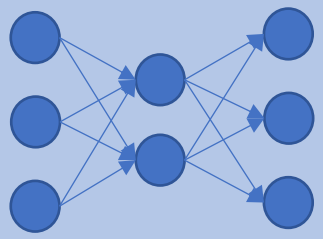
本教材の目次

深層学習

深層学習とはニューラルネットワークをモデルとする機械学習のことで以下のような関係性があります.

機械学習（Machine Learning, ML）
有限の観測データから背後に潜む規則を獲得.
教師あり学習・教師なし学習・強化学習という枠組みがある.

深層学習（Deep Learning, DL）
ニューラルネットワークを用いた機械学習.
AIにパラダイムシフトを起こした.



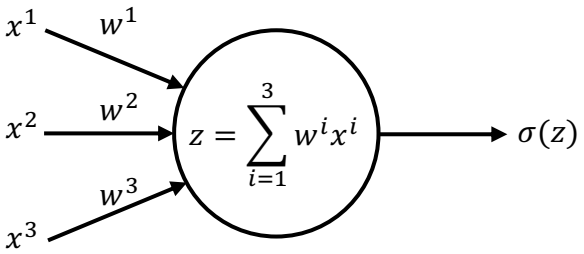
ニューラルネットワーク：
人の脳を模した学習機械.
画像・音声などの認識，自然言語処理，
深層強化学習などで用いられる.

ニューラルネットワークの原理

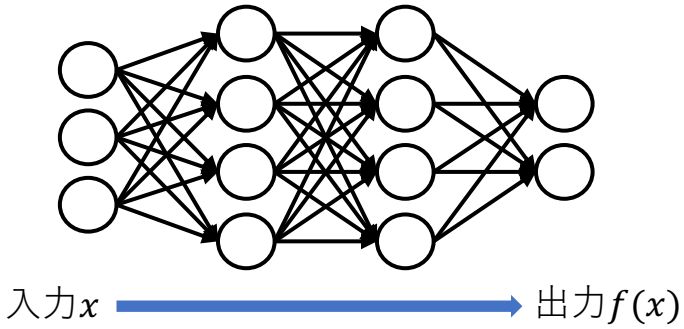
ニューラルネットワークは脳を模した機械学習モデルです.

ニューロン（ノード）が重みつきの枝を通じて接続し合いネットワークを構成します.

ニューロンは接続関係にあるニューロンから枝の重みに従い信号を受け取ったのち，活性化関数と呼ばれる適当な変換をかけ信号を出力します.



この信号の変換を多数のニューロンを用いて層状に積み重ね，入力データの複雑な変換を実現したモデルがディープニューラルネットワーク（DNN）です.

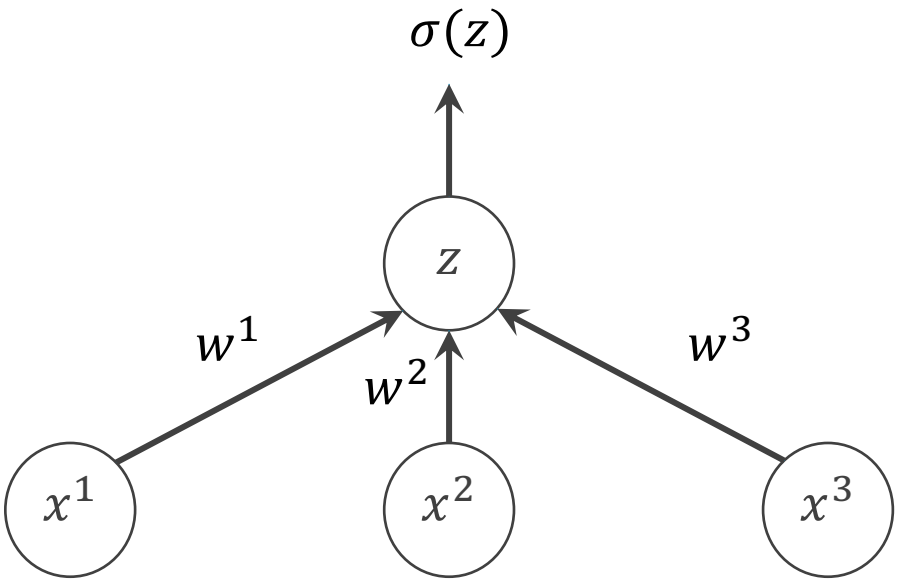


枝の重みをパラメータとして経験損失最小化を行いニューラルネットワークの学習を行います.

ニューロン

ニューラルネットを構成する最小単位.

各ニューロンは接続されたニューロンから枝の重みwをかけ総和をとったのちに活性化関数σを適用し出力値を定めます.



$$z = w^{\mathsf{T}} x = \sum_{i=1}^{d} w^i x^i .$$

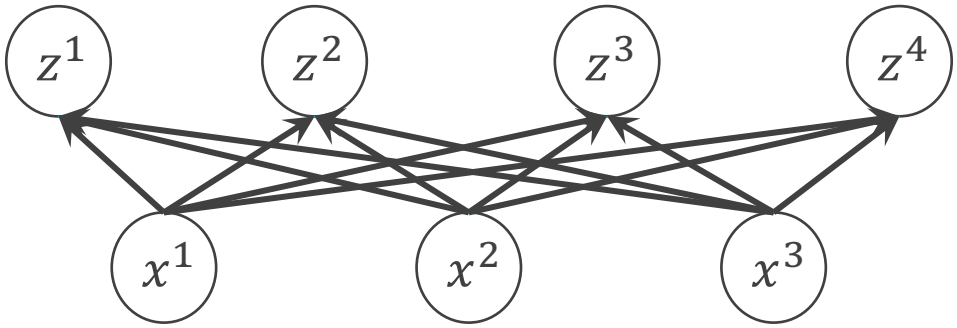
σ: 活性化関数.
ReLU, sigmoidと呼ばれる活性化関数などがよく用いられます. これらの活性化関数は次で定義されます.

$$\sigma(z) = \begin{cases} \max\{0, z\} \quad (\text{ReLU}), \\ \dfrac{1}{1+e^{-z}} \ (\text{sigmoid}). \end{cases}$$

入力 $x = \left(x^1, x^2, \ldots, x^d\right)^{\mathsf{T}}$.
重み $w = \left(w^1, w^2, \ldots, w^d\right)^{\mathsf{T}}$.

層

ニューロンを並列に束ね層を構成します.



ニューロンを接続する枝には重みパラメータWがあります.

入力ニューロンの値を$x = \left(x^1, x^2, \ldots, x^d\right)^\top$, 出力ニューロンの値を$z = \left(z^1, z^2, \ldots, z^h\right)^\top$とすれば, 重みパラメータ$W$は行数$h$, 列数$d$の行列であり

この層による入力xの変換は$z = \sigma(Wx)$とかけます.

注意：ここで活性化関数σは全出力ニューロンに適用されます.

このような接続関係がある層は全結合層と呼びます.

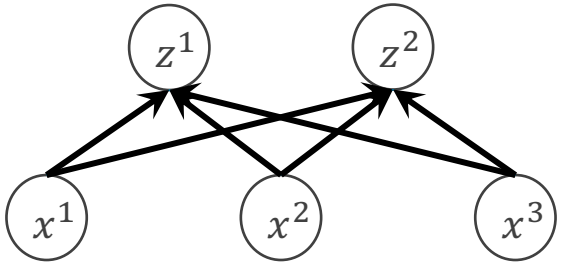
パラメータWは基本的に経験損失最小化による学習で調整されます.

ロジスティック回帰

ロジスティック回帰の詳しい説明は1-4「データ分析」を参照.

多クラス分類問題に対する線形ロジスティック回帰は最も単純なニューラルネットワークの学習問題とみなせます.

クラス数をCとしてラベルは $y = 1, \dots, C$ をとることにし入力データは $x = (x^1, x^2, \dots, x^d)^\top$ とします. ロジスティック回帰では行数C, 列数dのパラメータWとC次元のバイアスパラメータ$b = (b^1, b^2, \dots, b^C)^\top$により $z = f_{W,b}(x) = Wx + b$ という変換をします.



ロジスティック回帰では入出力ペア(x, y)に対して次の損失関数を定めます：

$$l\big(f_{W,b}(x), y\big) = -\log \frac{\exp(z^y)}{\sum_{k=1}^{C} \exp(z^k)}.$$

これは一層のニューラルネットワークをモデルとし交差エントロピー損失を損失関数とする問題設定に他なりません.

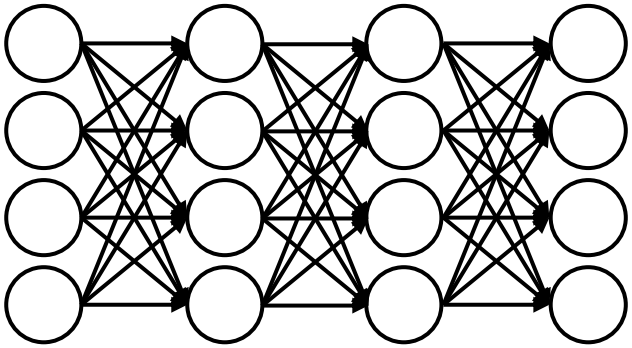
ディープニューラルネットワークの構造

ディープニューラルネットワークは層を直列に結合したモデルです.

非線形活性化関数を挟むことで複雑な非線形変換を実現します.

$$x \longrightarrow W_1 x \longrightarrow \sigma_1(W_1 x) \longrightarrow W_2\sigma_1(W_1 x) \longrightarrow \sigma_2(W_2\sigma_1(W_1 x))$$

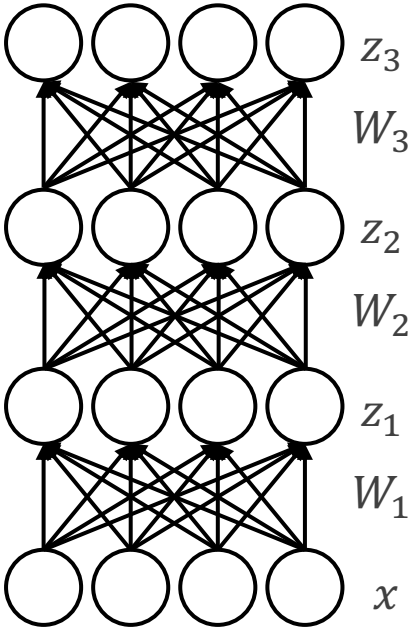
入力　線形変換　非線形変換　線形変換　非線形変換



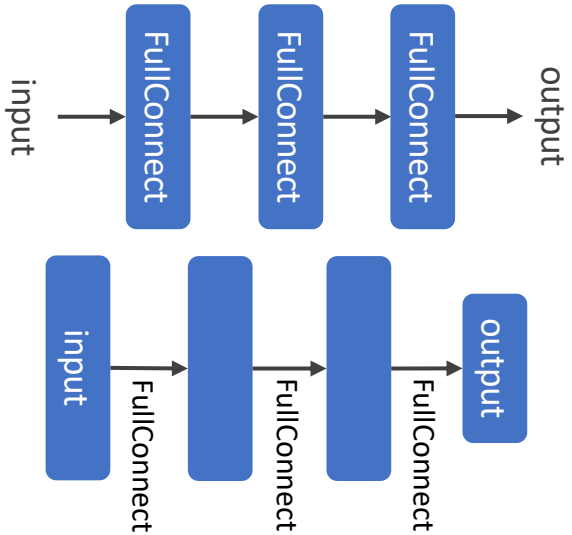
最終層には非線形活性化関数を適用しない場合が多いです.

そして最終層の値に対してタスクに応じた損失関数（交差エントロピー損失など）を適用し学習を実行します.

ディープニューラルネットワークの構造



層を重ねディープニューラルネットワークを構成.
左図：全結合層から成る３層ニューラルネットワーク.
簡易的に下図のように表現する場合もあります.



変換部分に注目した表現.

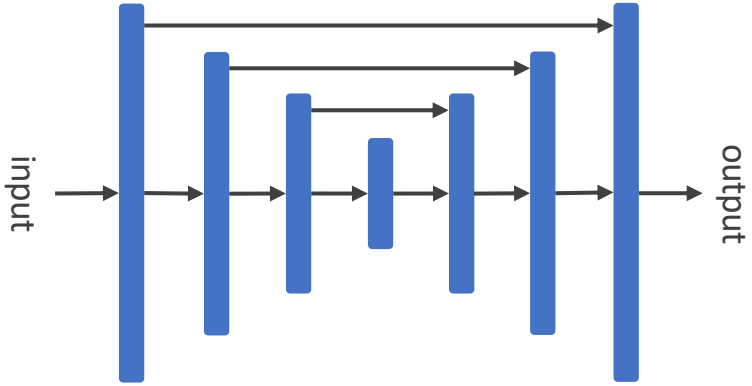
ニューロンの集まりに
注目した表現.

層は特定の用途・目的に応じデザインされ，種々のものがあります.

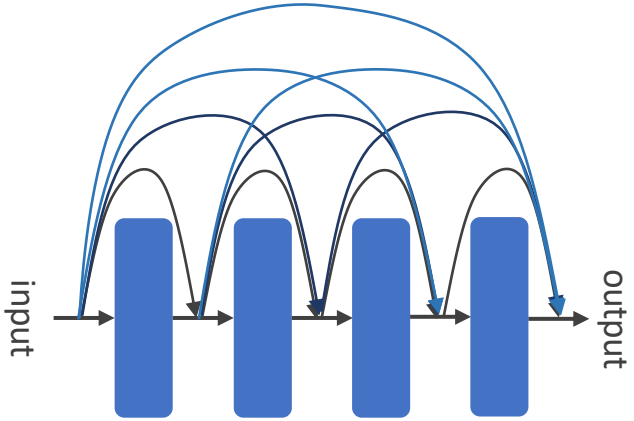
全結合層，ソフトマックス層，活性化層，畳み込み層，プーリング層，
LSTM層，バッチ正規化層，ドロップアウト層.

柔軟なモデリング

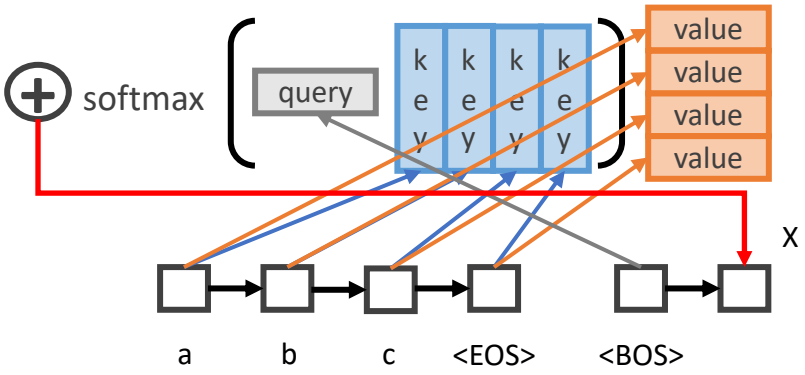
層をブロックとして組み合わせ自由自在なモデリングが可能です.
タスクに応じて直感的な依存関係を柔軟に記述することできます.



U-Net: 画像セグメンテーションで有効なモデル



DenseNet: 高精度な認識モデル



Attention: 入力(query)に対応する値(value)を返す辞書(keyとvalueのペア).

TransformerはAttention機構を組み込んだ機械翻訳モデル.

実世界で進む深層学習の応用と革新

タスク依存の直感的なモデリングはニューラルネットワークの精度を向上させる上での大きな利点です.

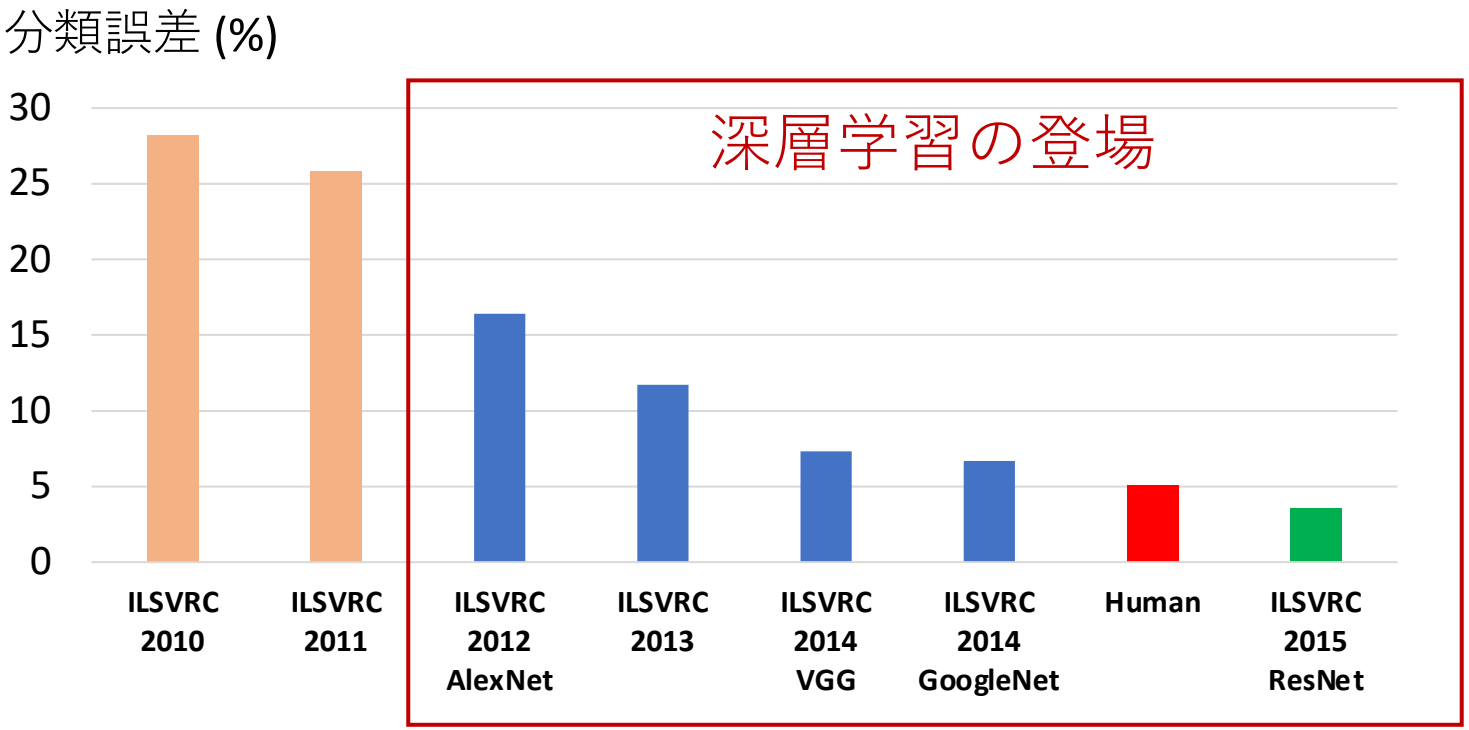
実際，タスクに応じてさまざまなニューラルネットワークモデルが提案され学習技法の発展とあわせて劇的な精度向上がもたらされました.

例えば以下のタスクで非常に優れたモデルや学習法が開発されました.

- **画像認識**
  畳み込みニューラルネットワーク（CNN）
- **自然言語処理**
  再帰型ニューラルネットワーク（RNN），Transformer
- **画像生成**
  敵対的生成ネットワーク（GAN）
- **音声生成**
  WaveNet

画像認識精度の向上

ImageNet（画像認識の大規模データセット）のコンペティションでの画像認識精度は深層学習時代のCNN登場以降，飛躍的に向上．

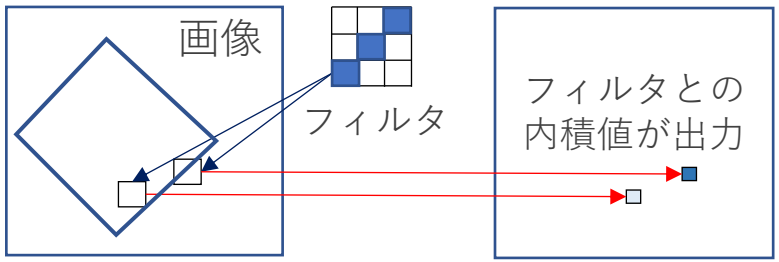


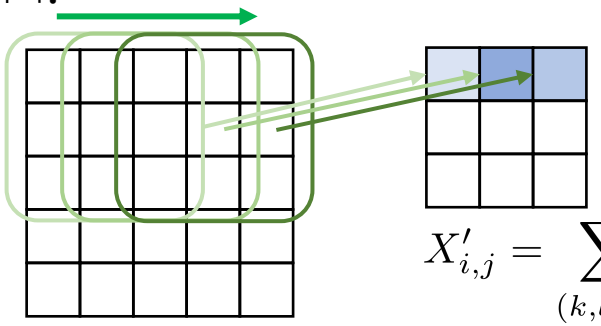
畳み込みニューラルネットワーク（CNN）

画像認識で高いパフォーマンスを発揮するニューラルネットワーク.

畳み込み層とプーリング層を交互に積み重ね最後に全結合層が接続されます.

- **畳み込み層**：入力画像のパターンの抽出.

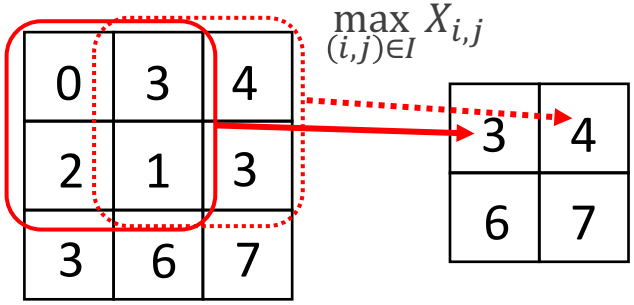




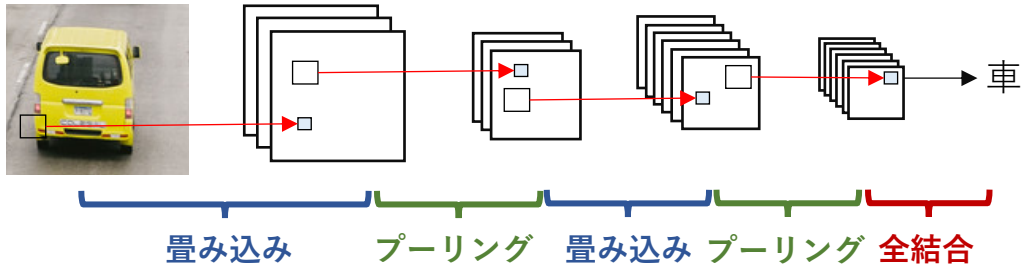
$$X'_{i,j} = \sum_{(k,l)\in I} X_{i+k,j+l} F_{k,l}$$

フィルタが畳み込み層のパラメータ．有用な情報が伝わるように学習される.

- **プーリング層**：平行移動不変性の獲得.



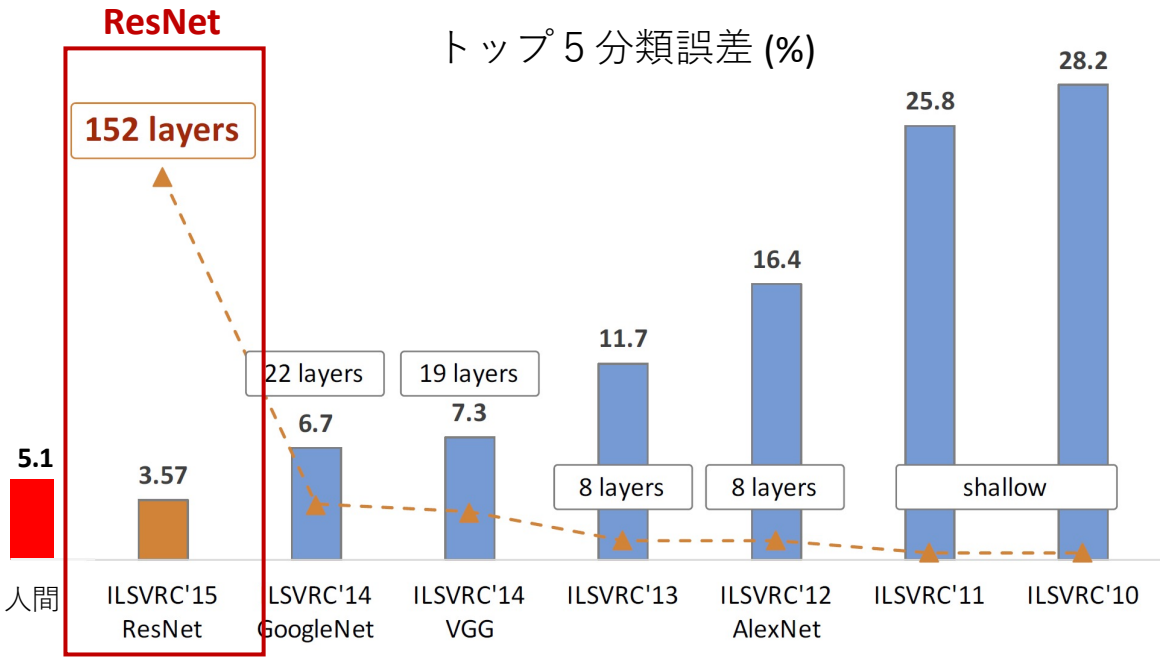
最大プーリングの例.
平均プーリングなどもある.

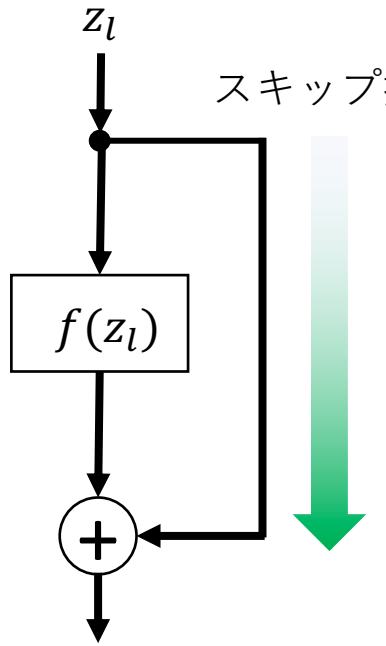


畳み込み層が2つプーリング層が2つからなるCNN.

Residual Network (ResNet)

従来，非常に深いネットワークの学習は困難でしたがスキップ接続という機構を備えたResNetの登場により巨大ネットワークの学習も可能となりました．それに伴い画像認識精度もさらに向上しました．





$$z_{l+1} = z_l + f(z_l)$$

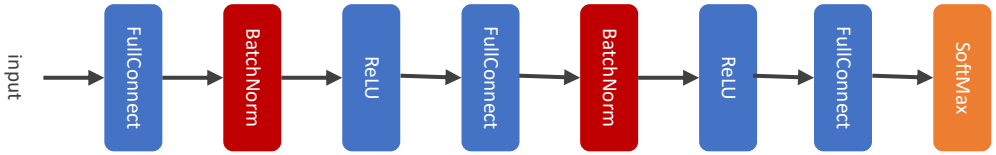
ResNetの各層の情報変換の様子．直前の層の値を直接受け取るパス（スキップ接続）がある．

さまざまな技法

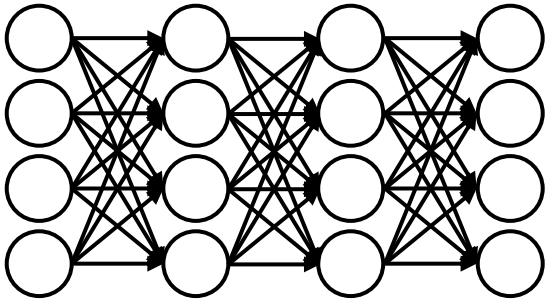
深層学習では精度向上のためのさまざまな技法が開発されました.

とくにバッチ正規化やドロップアウトは非常に有効です.

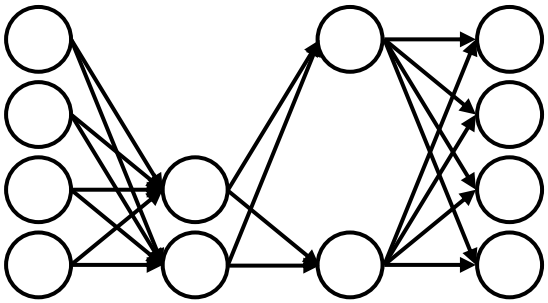
- バッチ正規化
  訓練時と予測時で各層での入力分布が異なることを共変量シフトと呼びます. 共変量シフトに伴う精度劣化を防ぐ技法がバッチ正規化で各層にバッチ正規化層を挿入することで実現されます.



- ドロップアウト
  訓練時にランダムにニューロンを削除しながら学習する正則化手法です. 汎化性能の向上に有効です.



通常時のニューラルネットワーク



ドロップアウト時のニューラルネットワーク

経験損失最小化

深層学習はニューラルネットワークモデルを用いた機械学習です.

ですので基本的に経験損失最小化問題を解きます：

$$\min_{W}\left\{L(W)=\sum_{i=1}^{n} l(f_W(x_i),y_i)\right\}.$$

損失関数lは二乗損失（回帰）や交差エントロピー損失（分類）を用います.

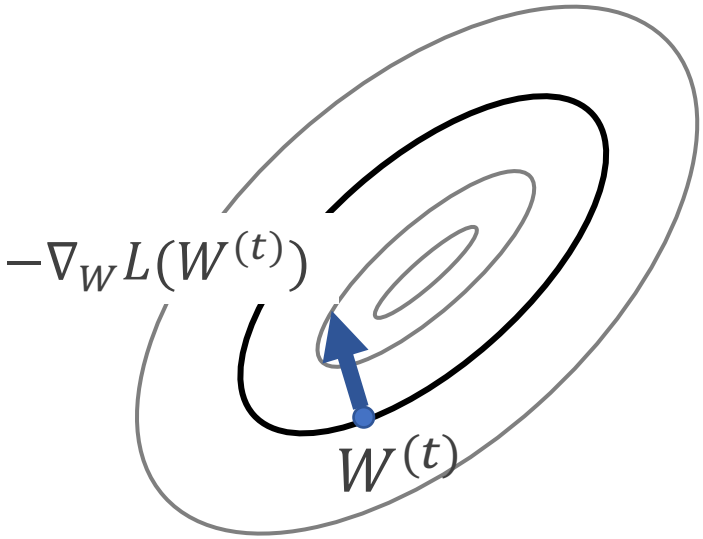
この最適化問題は確率的勾配降下法やその改良版であるADAMという手法で解くことが一般的です.

確率的勾配降下法は最急降下法を確率化したもので，機械学習において非常に有効な最適化手法です.

最急降下法

最急降下法は反復法の一種で，適当に決めた初期パラメータ$W^{(1)}$から逐次的に以下の手続きでパラメータを更新します：

$$W^{(t+1)} = W^{(t)} - \eta \nabla_W L\left(W^{(t)}\right).$$



ここで$\nabla_W L(W^{(t)})$は$W^{(t)}$における経験損失関数Lの勾配です．ηはステップサイズや学習率とよばれるハイパーパラメータで，パラメータを変化させる幅を調整します．

勾配は関数の等高線と直交しますので負の勾配は関数を局所的に減少させる方向になります．

従ってηがある程度小さいとき，最急降下法により損失関数が減少することが分かります．

ニューラルネットワークは行列による線形変換を繰り返すモデルでした．そのためパラメータについての勾配は微分における連鎖率というルールで計算されます．これは深層学習と線形代数/微分積分との関係性を指していて，深層学習を正しく理解するには微分積分や線形代数がある程度必要であるといえます．

確率的勾配降下法

経験損失関数はデータ数分の損失の和で定義されていました：

$$L(W) = \frac{1}{n}\sum_{i=1}^{n} l(f_W(x_i), y_i)\,.$$

従って最急降下法のパラメータの更新規則は次のように書けます：

$$W^{(t+1)} = W^{(t)} - \frac{\eta}{n}\sum_{i=1}^{n} \frac{\partial l(f_W(x_i), y_i)}{\partial W}.$$

全データでの計算が必要で遅い

ここでパラメータの更新の度に全データ分の計算が必要となり非効率です.

そこで各tにおいてn個のデータから乱択された1データ(x_{i_t}, y_{i_t})のみを用いた
近似的な勾配でパラメータを更新する手法が確率的勾配降下法です：

$$W^{(t+1)} = W^{(t)} - \eta \frac{\partial l(f_W(x_{i_t}), y_{i_t})}{\partial W}.$$

これにより計算効率が改善され大規模データに対する深層学習が可能となります.

深層学習を実行する際の手続き

深層学習を行う一連の手続きをまとめると概ね以下のようになります.

1. データを用意,
2. タスクに合わせてネットワーク構造をモデリング,
3. さまざまな技法や最適化手法を選択し学習を実行.

しかしながら大規模データセットの用意や，1からのモデリングおよび学習には非常に大きなリソースを要するという問題があります.

そこで画像認識などの汎用タスクでは色々な類似タスクで有効な学習用データと学習済みモデルが配布されています． これらをうまく活用したり，微調整することで効率的にタスクを解くことができます.

例えば以下のようなモデルは広く使われています.

画像認識：VGG, ResNet, DenseNet

オブジェクト検出：Yolo, R-CNN

画像セグメンテーション：U-Net

画像生成：BigGAN

深層学習の実行方法

深層学習はライブラリが整備されていて簡便に試すことができます.

例えばKerasというライブラリを用いて以下のようにディープニューラルネットワークを学習できます.

```
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

必要なライブラリをインポート.

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
        optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999),
        metrics=['accuracy'])
model.summary()
```

ニューラルネットワークモデル, 損失関数, 最適化法の指定.

```
epochs = 5
batch_size = 128
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
            verbose=1, validation_data=(x_test, y_test))
```

最適化を実行.

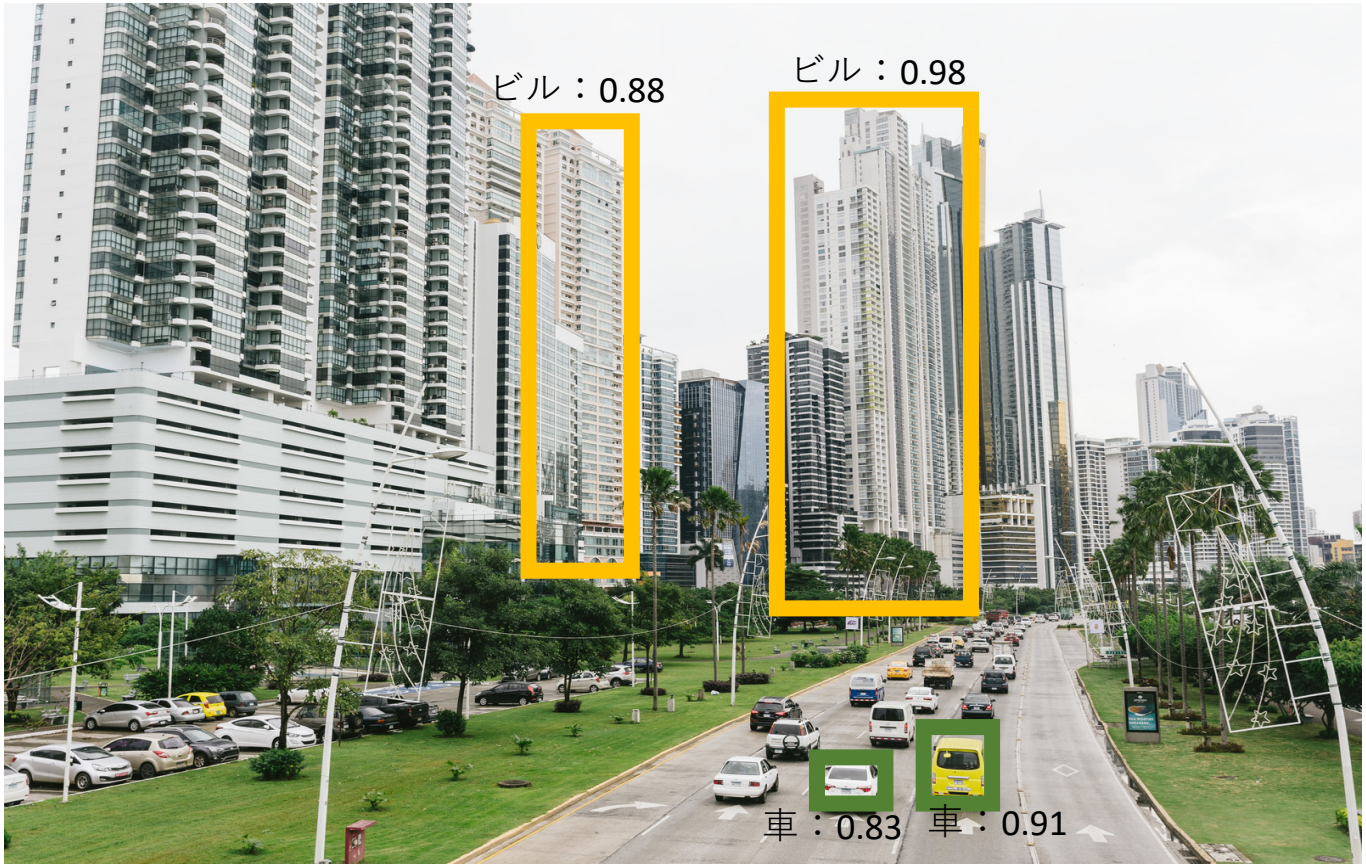
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

精度評価.

事例：物体検出

画像認識より一段むずかしい問題設定.

どこに何が写っているかを検出します.



「どこに」＋「なにが」

事例：マスキング

物体検出より詳しくピクセル単位で認識する.

ピクセルが何を構成しているか予測します.

事例：画像生成

敵対的生成ネットワーク(GAN)により高精細な画像の生成も可能です.

GANでは識別ネットワークと生成ネットワークという二つのネットワークが以下の指針で敵対的に交互に学習されます.

1. **識別ネットワーク**：本物の画像か生成ネットワークが生成した画像かを識別.
2. **生成ネットワーク**：識別ネットワークを騙すような画像を生成.

この敵対的学習法により次第に高精細画像が生成されるようになります.