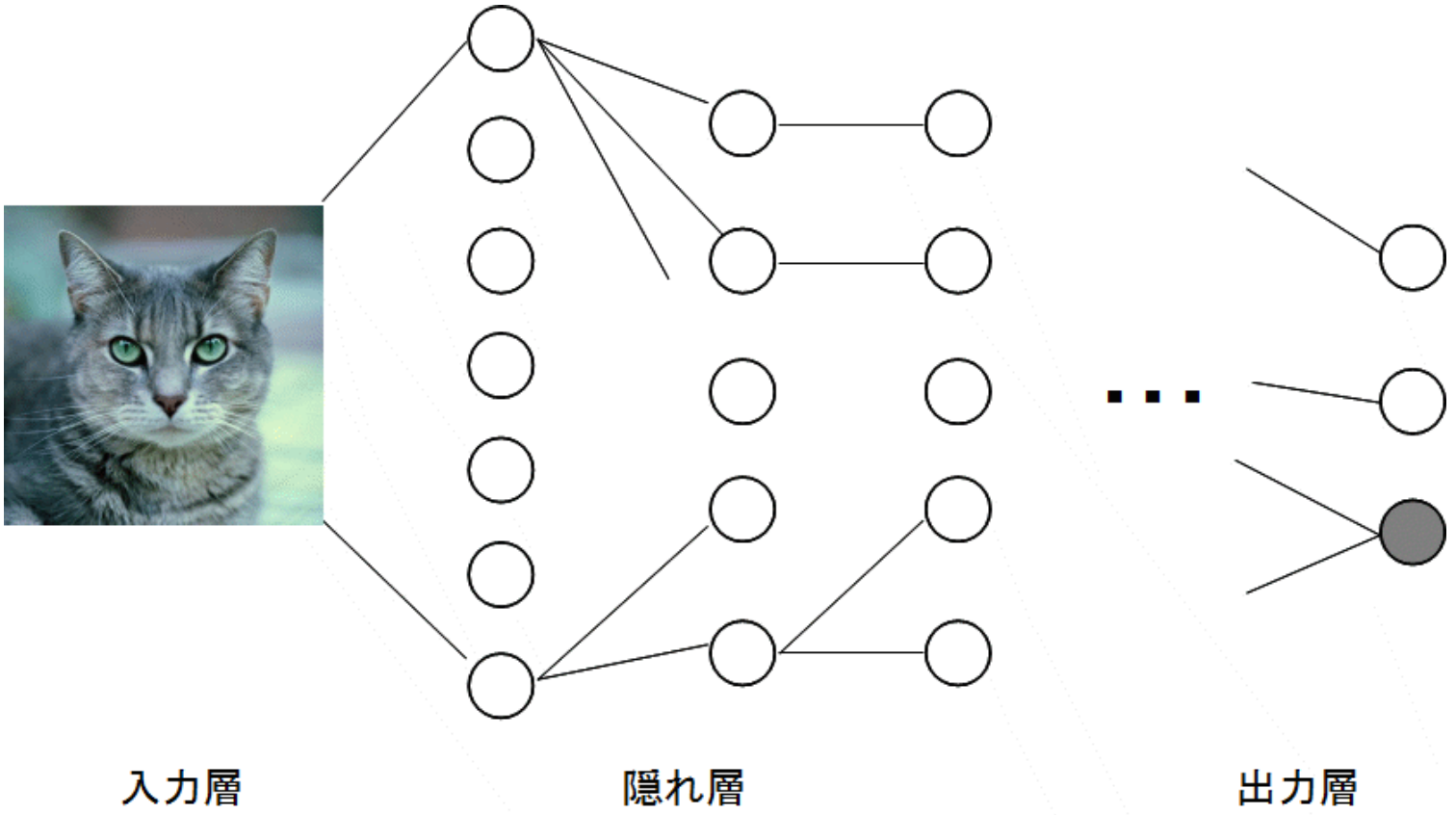
# Section 2

- 深層学習 (15章)

# 15 章 深層学習

## 15.1 深層学習とは

- 深層学習の定義のひとつ
  - 表現学習：抽出する特徴も学習する



# 15.1 深層学習とは

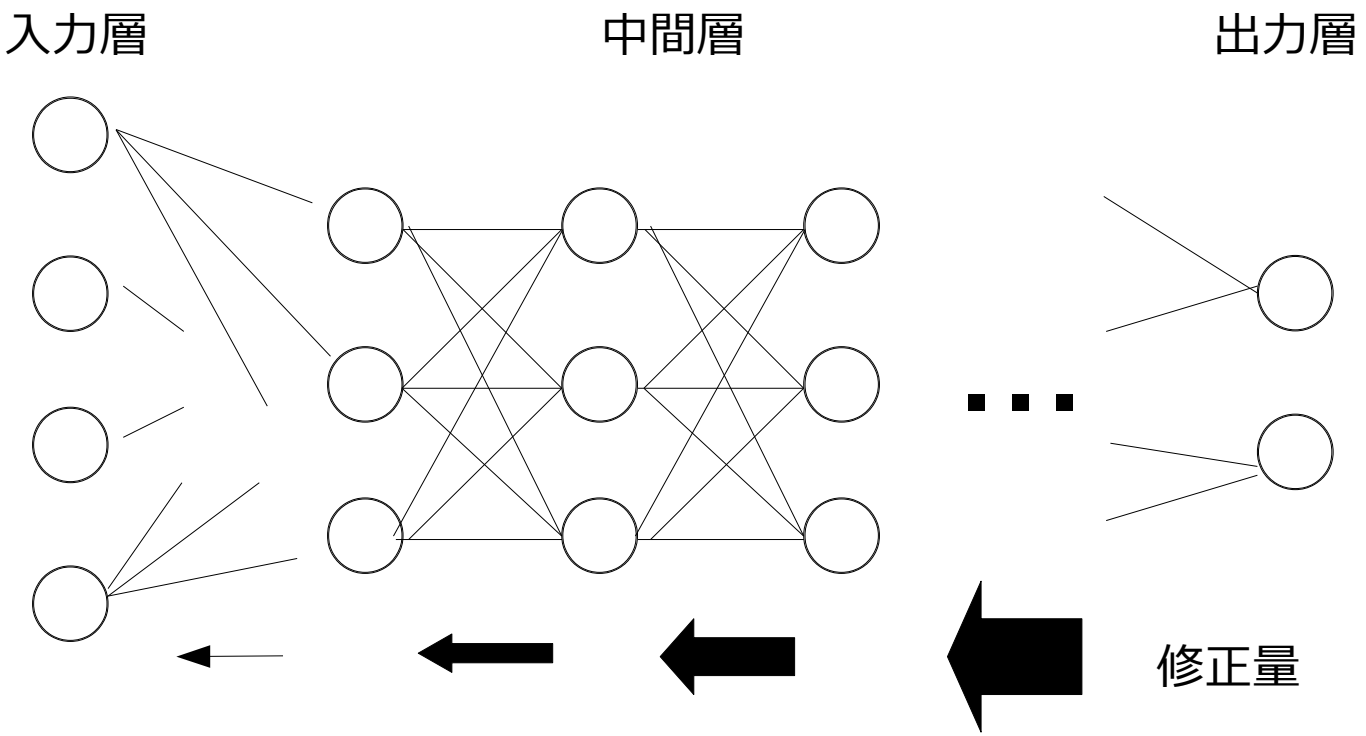
## 単純なマルチレイヤーパーセプトロンとの違い

- 多階層学習における工夫
  - 事前学習
  - 活性化関数の工夫
  - 過学習の回避：ドロップアウト
- 問題に応じたネットワーク構造の工夫
  - 畳み込みネットワーク
  - リカレントネットワーク

# 15.2 多階層ニューラルネットワークの学習

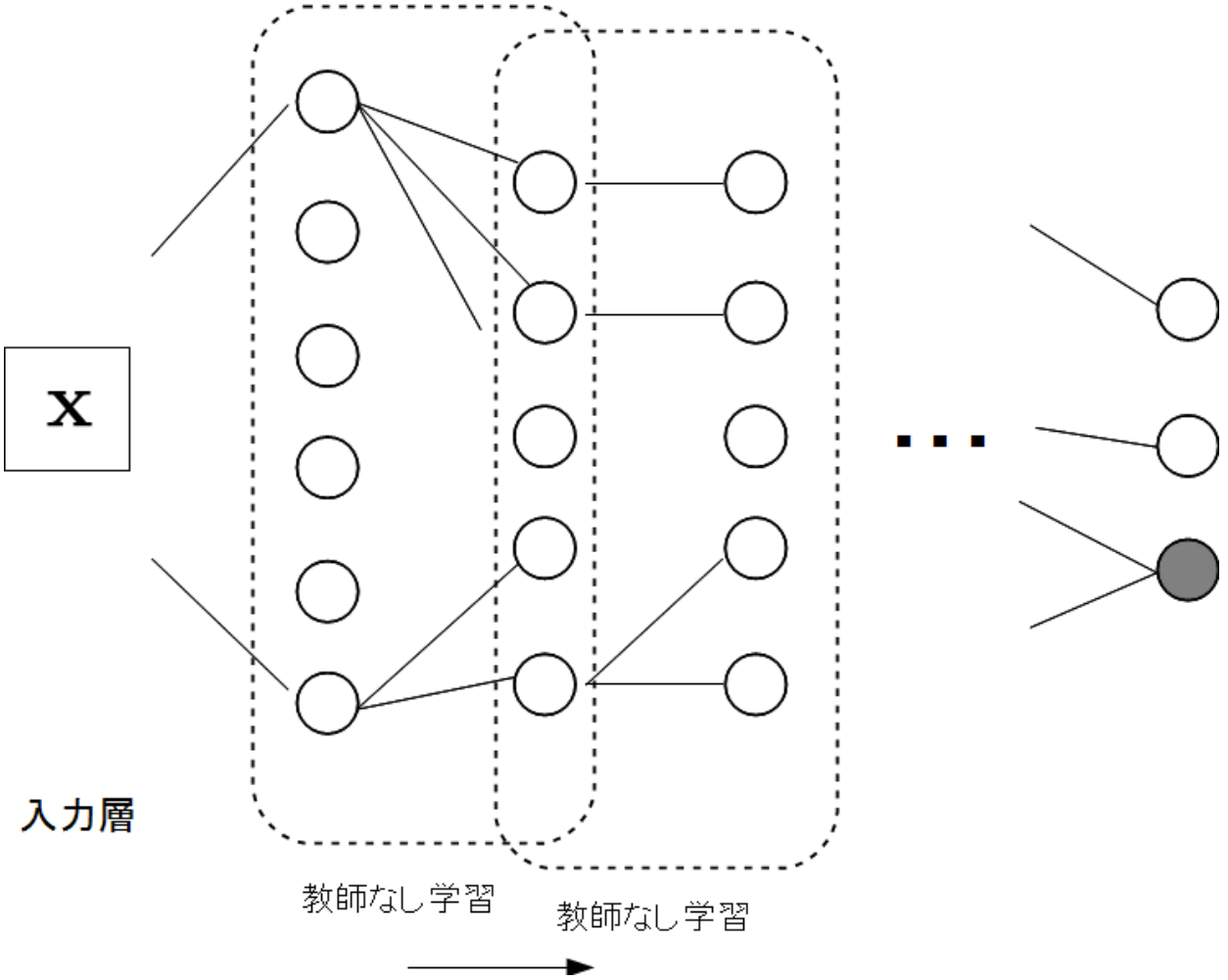
- 多階層における誤差逆伝播法の問題点
  - 修正量が消失／発散する

順方向：非線形　　逆方向：線形



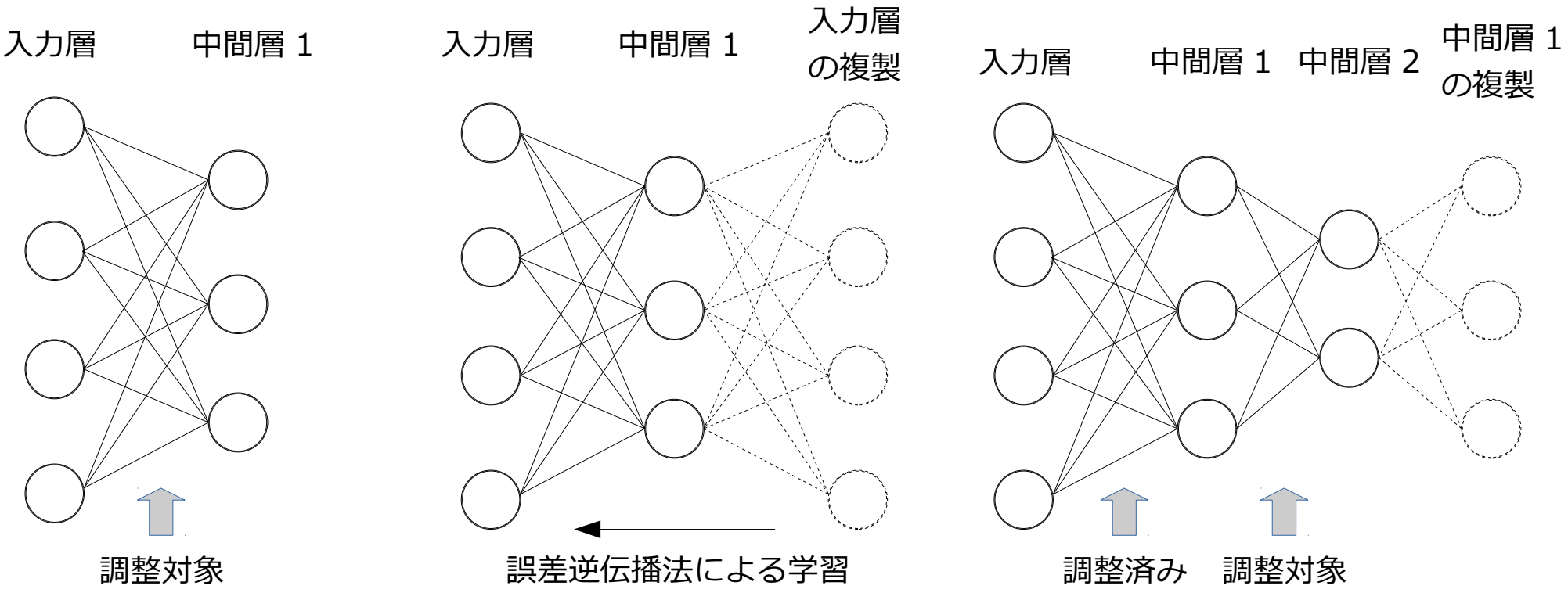
# 15.2 多階層ニューラルネットワークの学習

- 事前学習法のアイディア
  - 深層学習における初期パラメータ学習



# 15.3 Autoencoder

- autoencoder のアイディア：自己写像を行う




(a) 事前調整対象の重み

(b) オートエンコーダによる復元学習

(c) 1 階層上の事前調整

# 15.3 Autoencoder

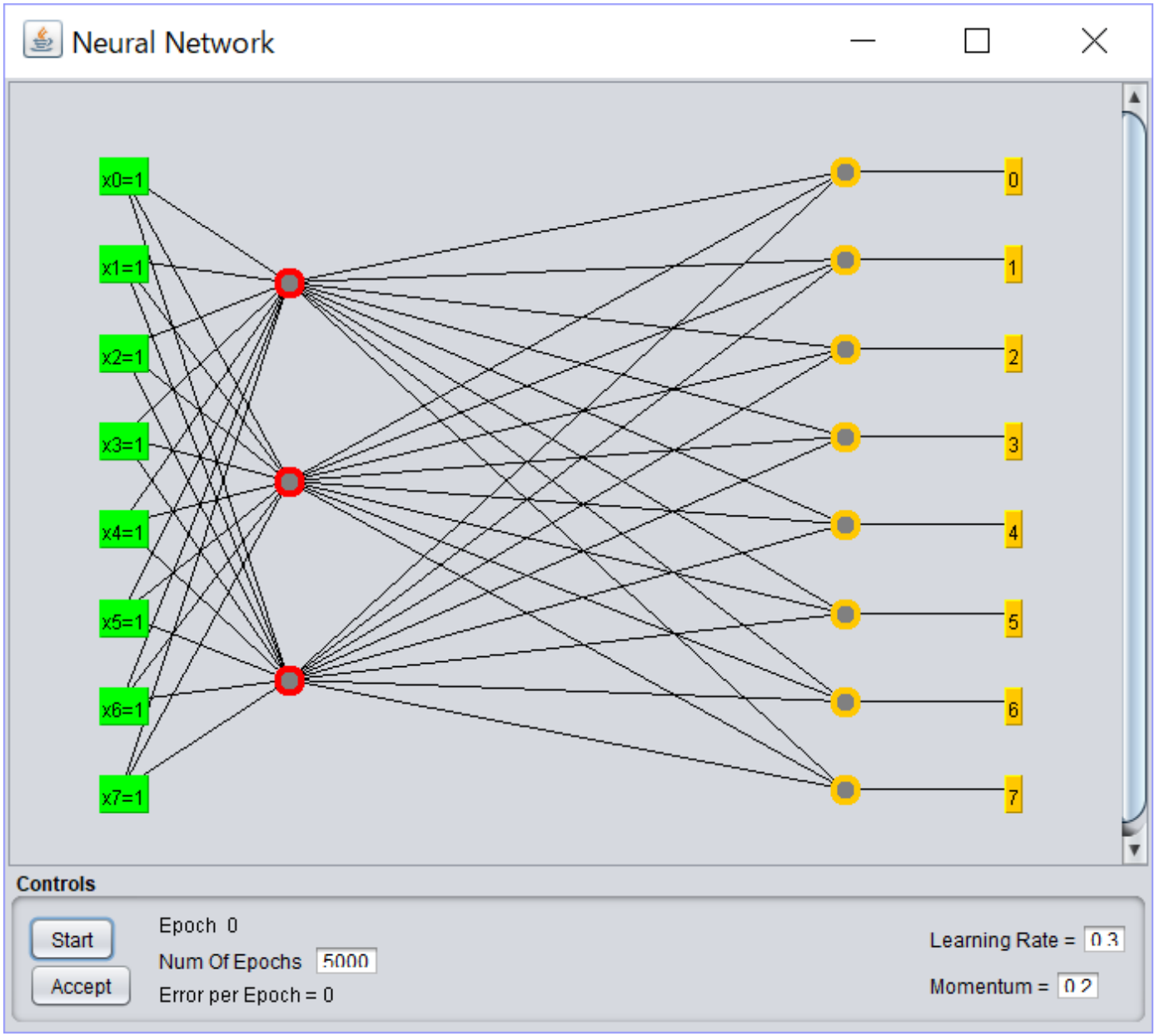
- 例題 15.1
  - ２進数の概念を獲得する autoencoder の実現
  - 入力： 0 ～ 7 の数字の one-hot 表現
  - 中間層：ユニット数 3
  - 出力： 0 ～ 7 の 8 クラス
  - 学習回数： 5000 回

```
@relation autoencoder

@attribute x0 {0,1}
@attribute x1 {0,1}
...
@attribute x7 {0,1}
@attribute class
    {0,1,2,3,4,5,6,7}

@data
1,0,0,0,0,0,0,0,0
0,1,0,0,0,0,0,0,1
0,0,1,0,0,0,0,0,2
0,0,0,1,0,0,0,0,3
0,0,0,0,1,0,0,0,4
0,0,0,0,0,1,0,0,5
0,0,0,0,0,0,1,0,6
0,0,0,0,0,0,0,1,7
```

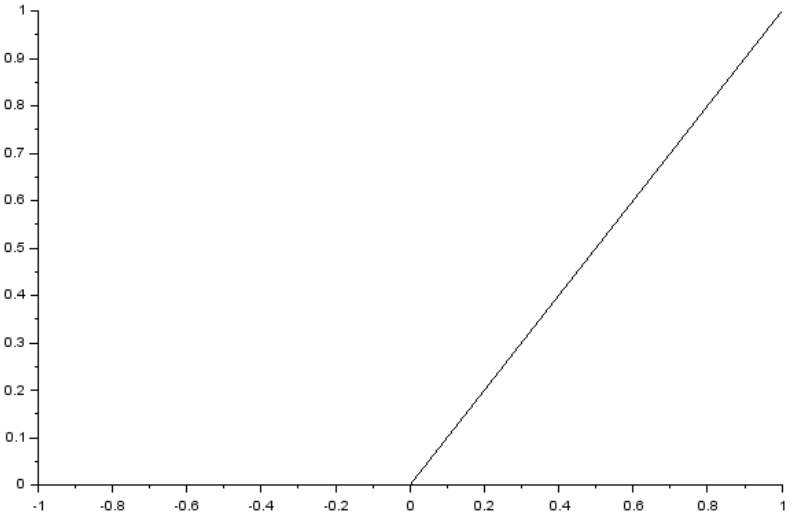
# 15.3 Autoencoder



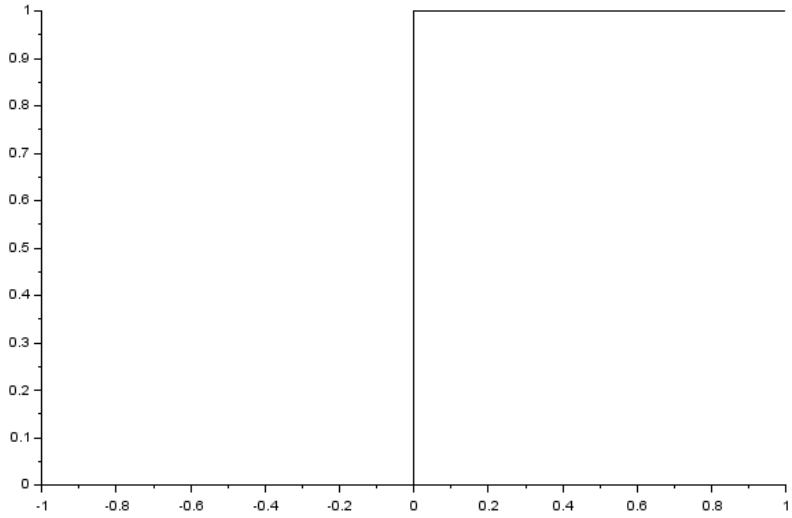
# 多階層学習における工夫

- 活性化関数を rectified linear 関数に ➡ RELU

$$f(x) = \max(0, x)$$




(a) rectified linear 関数

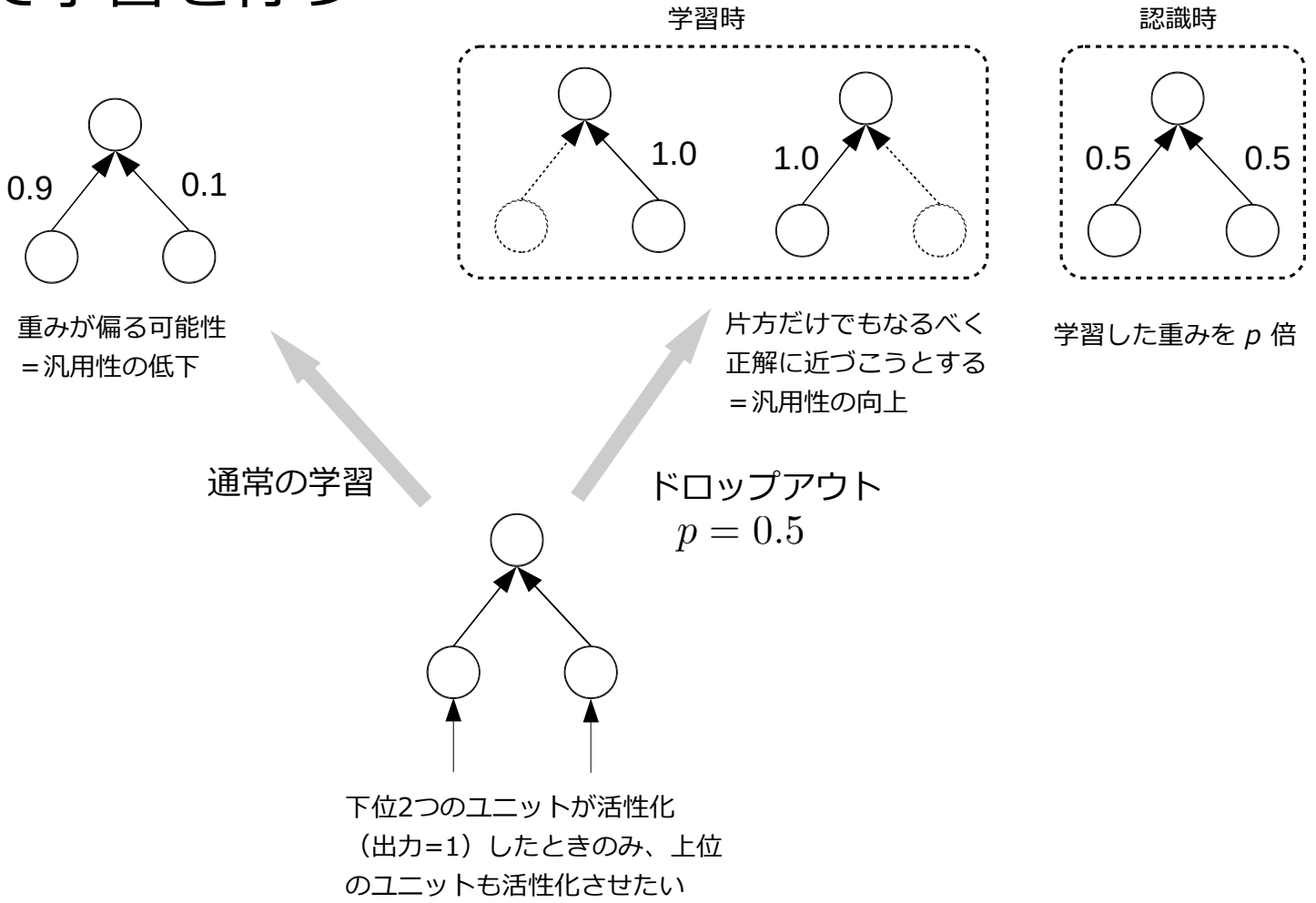



(b) (a) の導関数

- RELU の利点
  - 誤差消失が起こりにくい
  - 0 を出力するユニットが多くなる

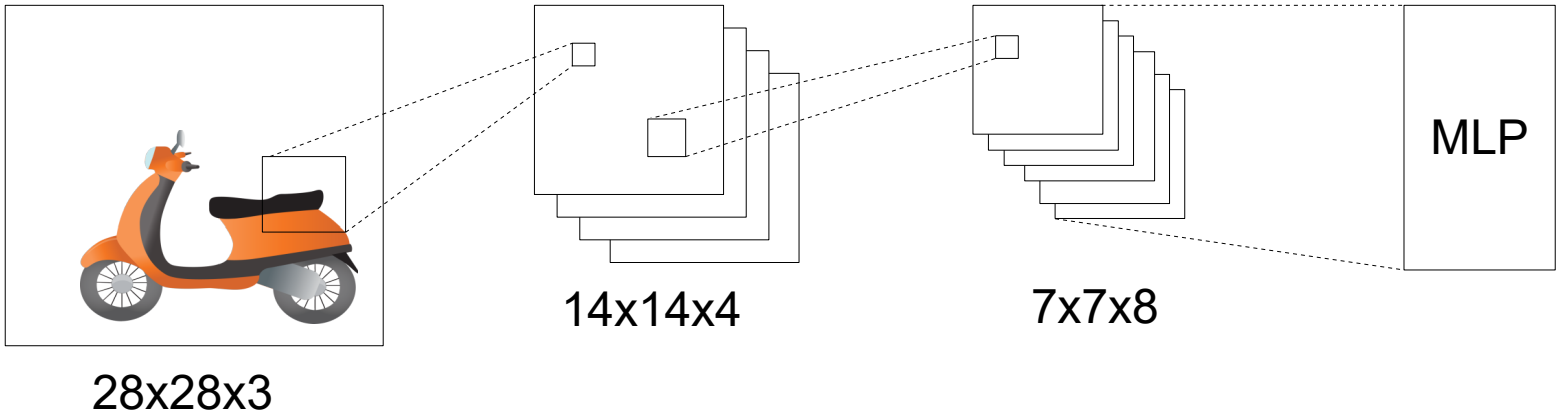
# 多階層学習における工夫

- 過学習の回避
  - ドロップアウト：ランダムに一定割合のユニットを消して学習を行う



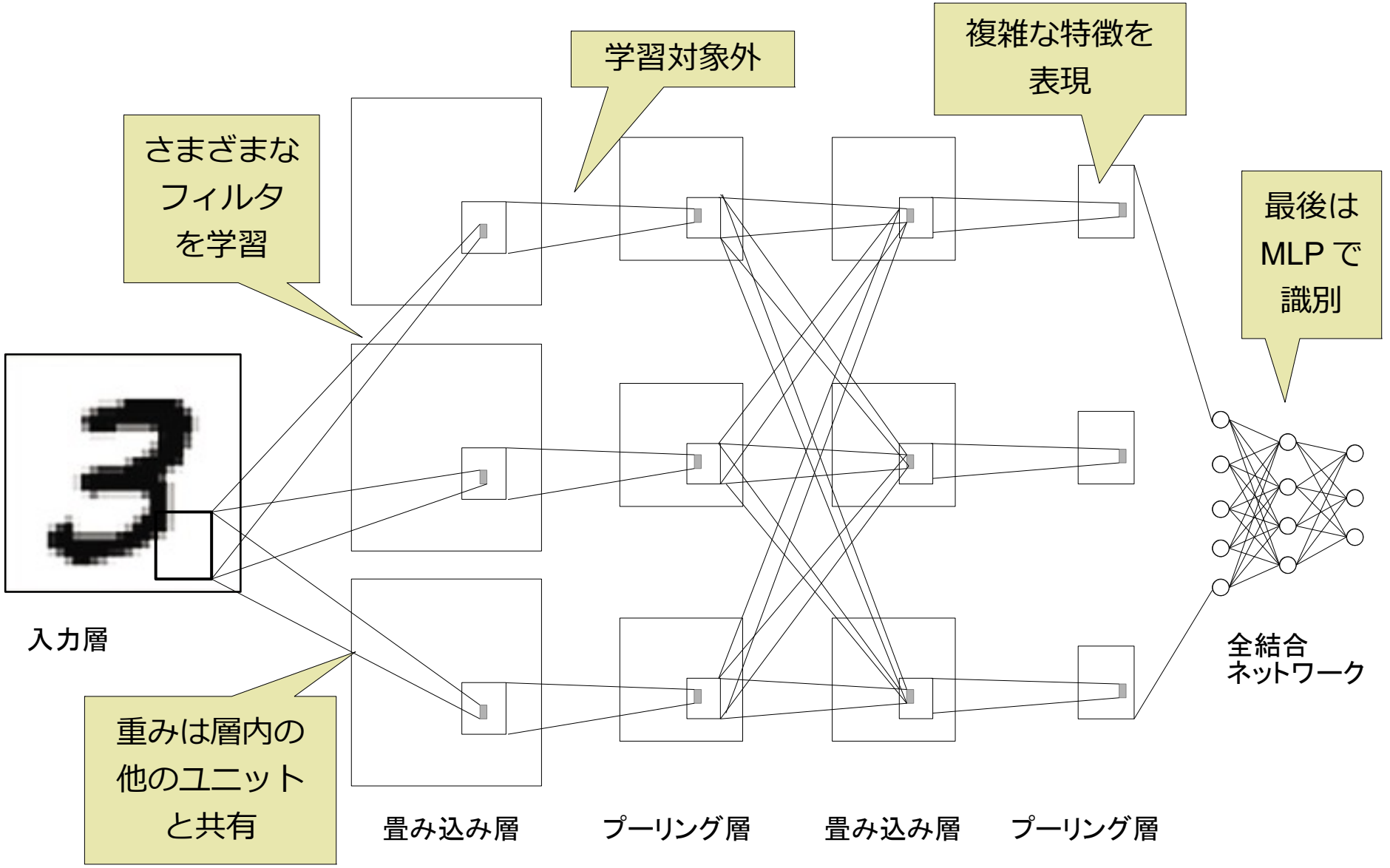
# 畳み込みニューラルネットワーク

- 畳み込みニューラルネットワークの構造
  - 畳み込み層とプーリング層を交互に重ねる
    - 畳み込み層はフィルタの画素数・ずらす画素数・チャネル数の情報からなる
  - 最後は通常の MLP （ Relu+softmax ）



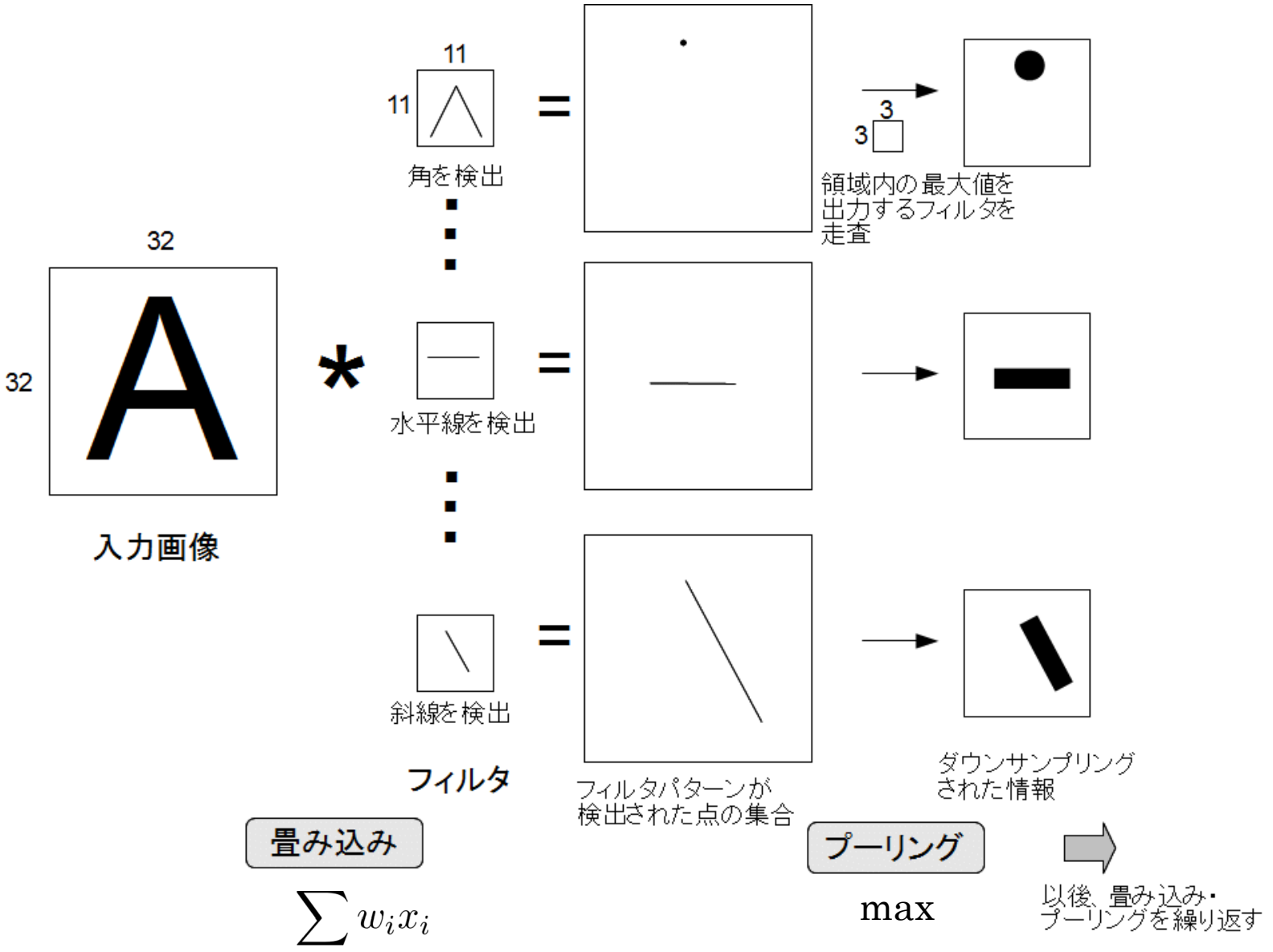
# 畳み込みニューラルネットワーク

- 畳み込みニューラルネットワークにおける学習



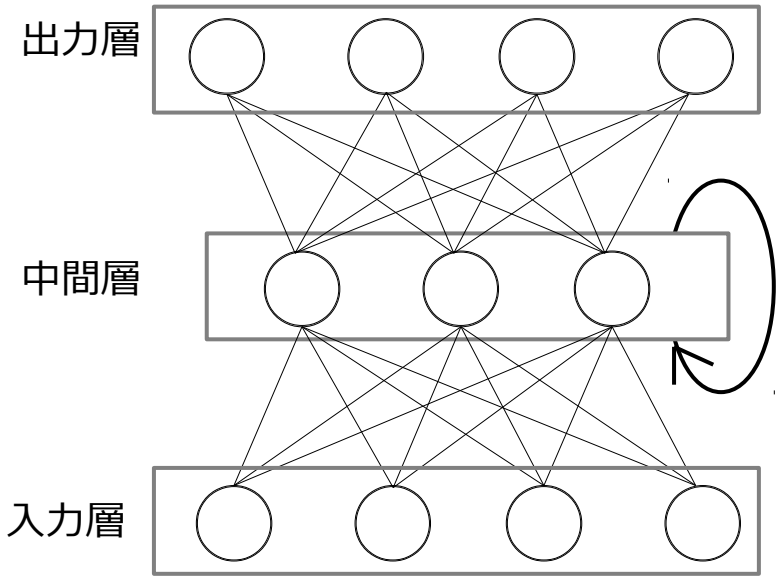
# 畳み込みニューラルネットワーク

- 畳み込みニューラルネットワークの演算

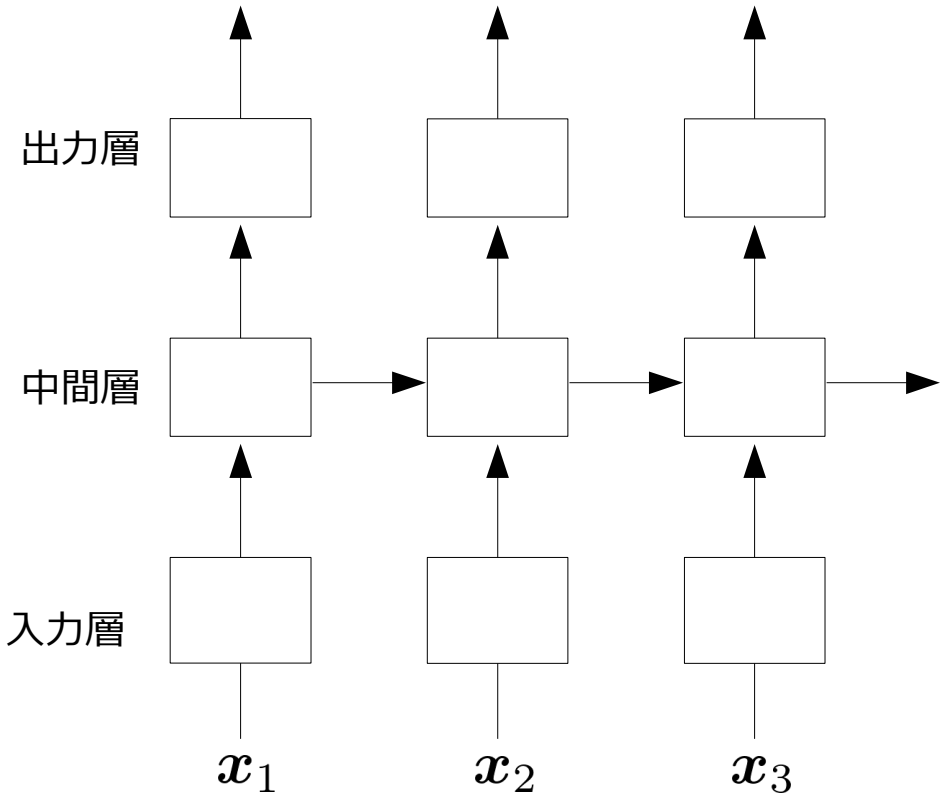


# リカレントニューラルネットワーク

- 時系列信号の認識や自然言語処理に適する



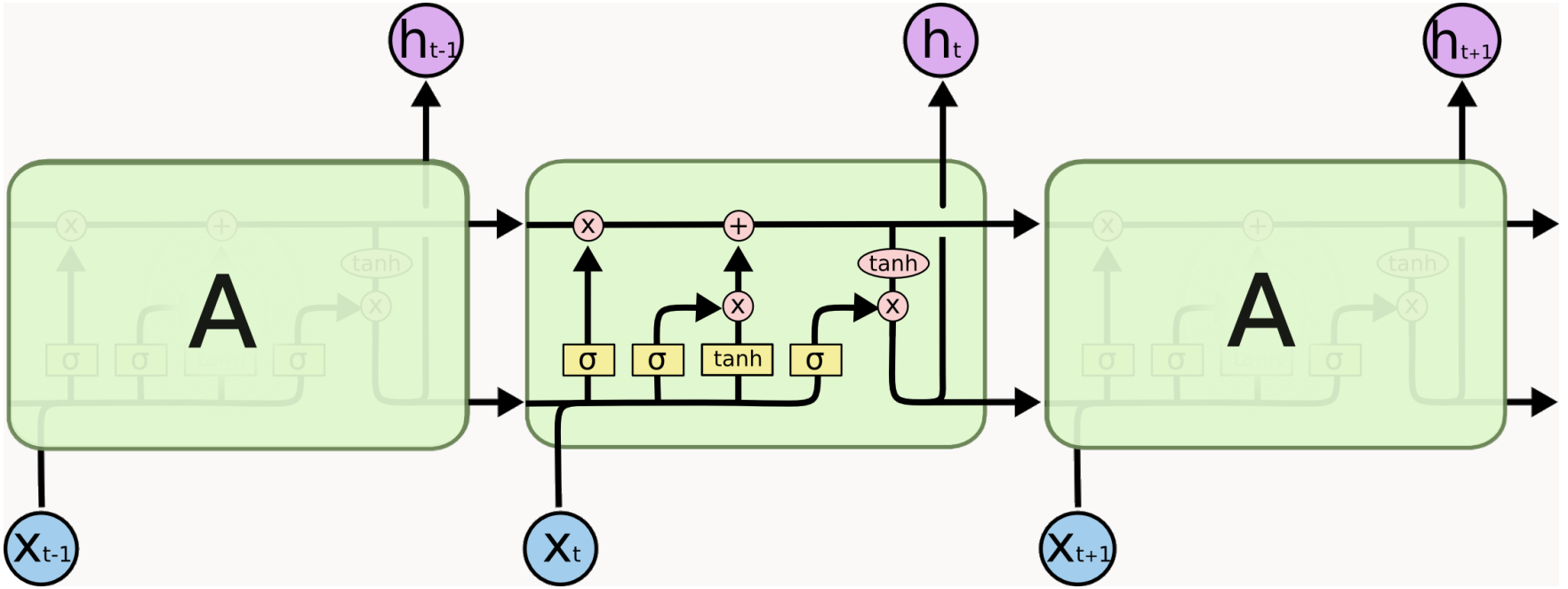

(a) リカレントニューラルネットワーク




(b) 帰還路を時間方向に展開

# リカレントニューラルネットワーク

- LSTM (long short-term memory)
  - いくつかのゲートからなる内部構造をもつユニット
    - ゲート：選択的に情報を通すメカニズム




  - 参考サイト

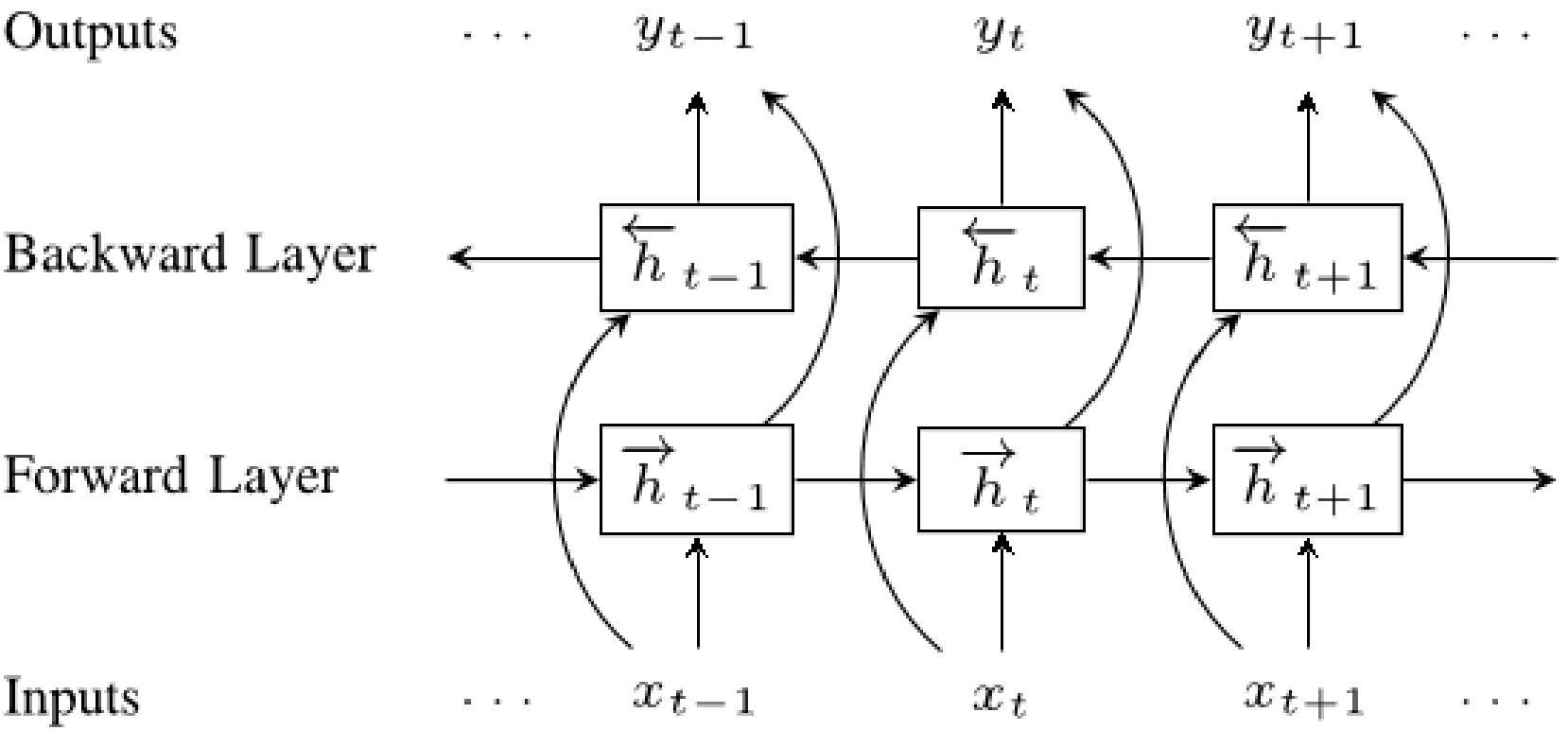
    http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# リカレントニューラルネットワーク

- LSTM のゲート
  - 忘却ゲート：セルの内容を捨てるかどうか
    - 例）言語モデルにおいて、新たな主語が現れた場合、古い主語の性別は捨てる
  - 入力ゲート：セルの内容のどの部分を更新するか
    - 例）古い主語の性別を新たな主語の性別で置き換える
  - 出力ゲート：セルの内容のどの部分を出力するか
    - 例）主語に続く動詞の形を決めるために、主語の単複を出力

# リカレントニューラルネットワーク

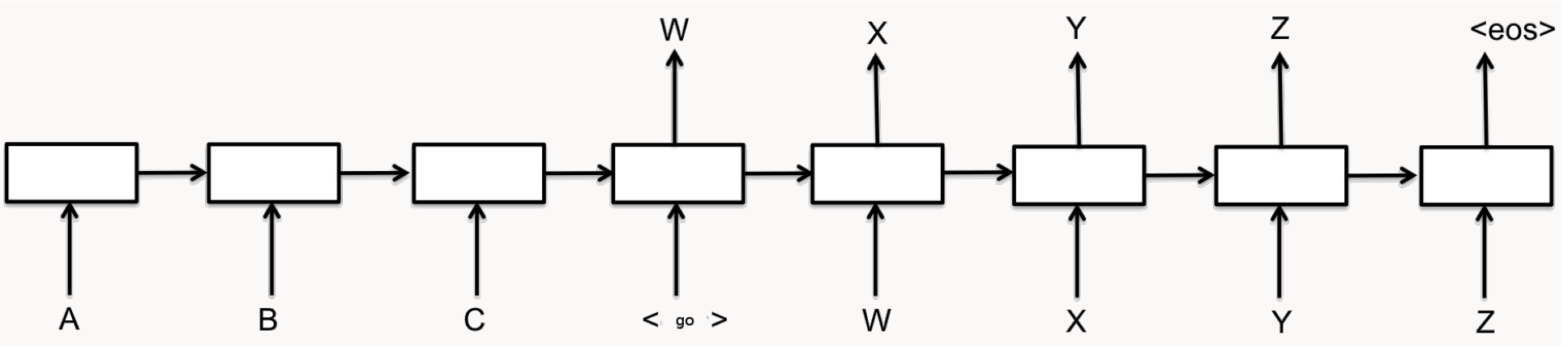
- Bidirectional RNN
  - 過去だけでなく、未来の情報も用いて出力を計算




He, L., Qian, Y., Soong, F.K., Wang, P., & Zhao, H. (2015). A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. CoRR, abs/1511.00215.

# リカレントニューラルネットワーク

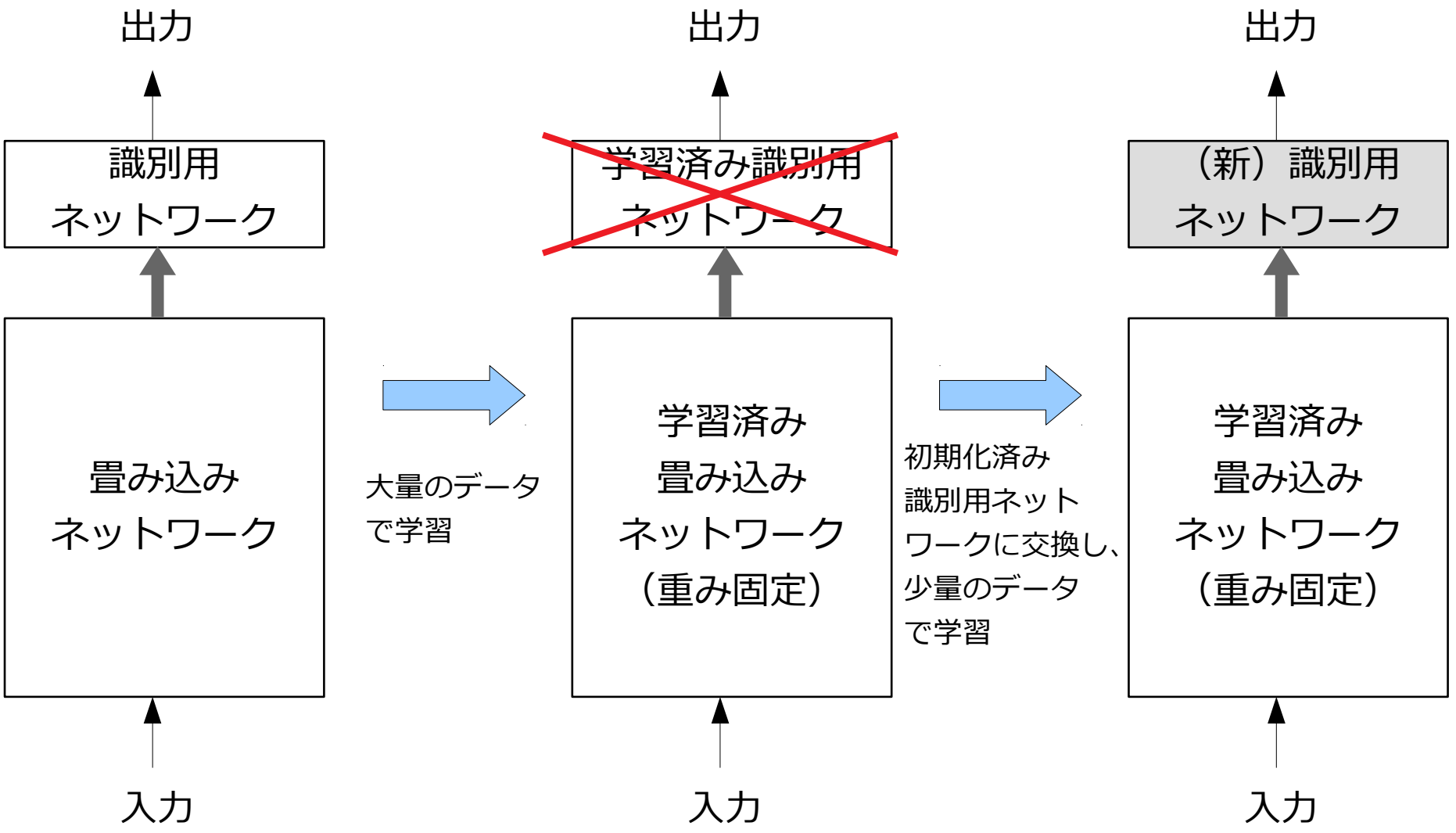
- Encoder-Decoder
  - 入力の内容をひとつの表現にまとめて、そこから出力を生成



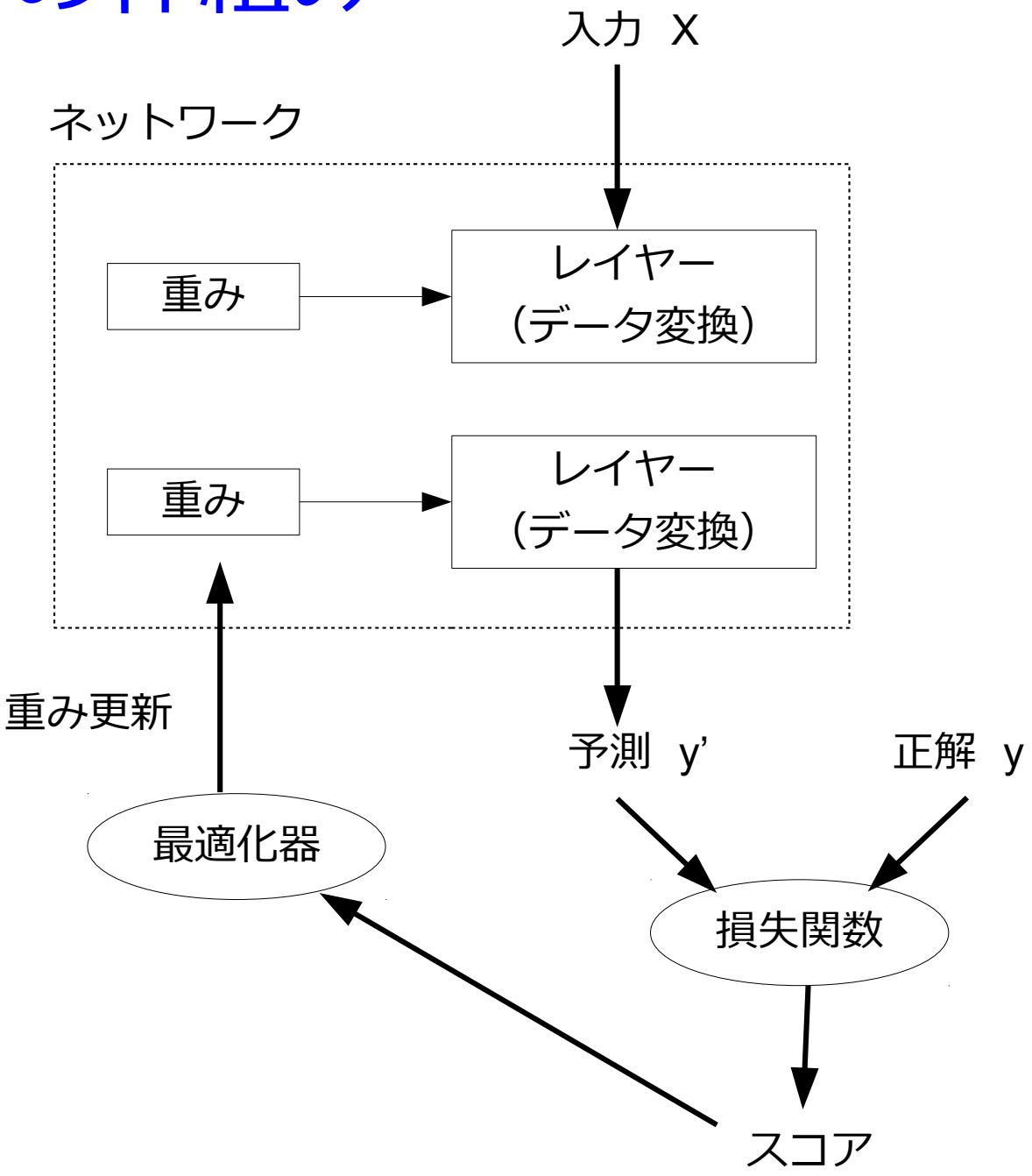

arXiv:1406.1078

# 転移学習

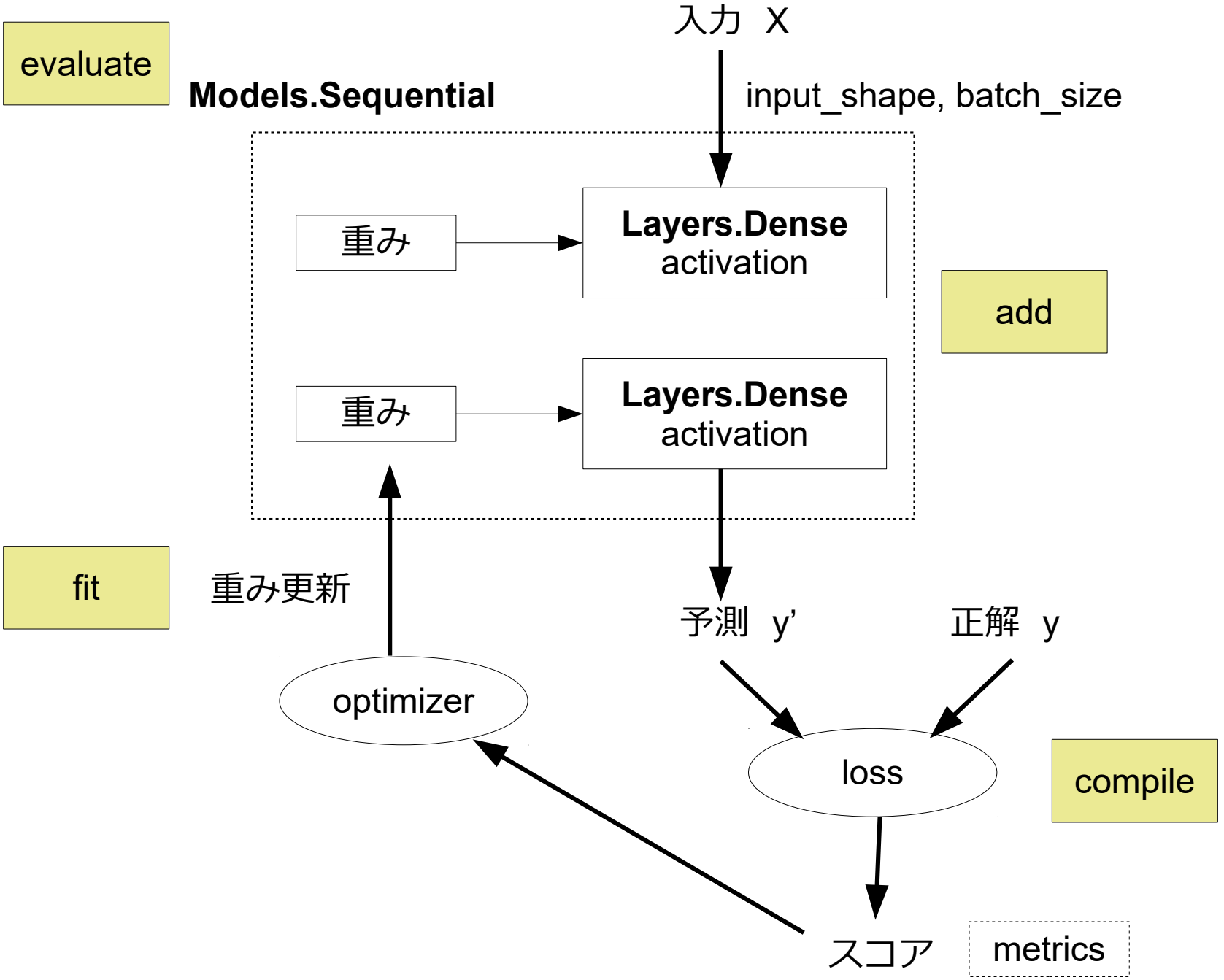
- 少ないデータ量でも DNN が活用できる可能性



# 深層学習の枠組み


入力 x
ネットワーク
重み
レイヤー
(データ変換)
重み
レイヤー
(データ変換)
重み更新
予測 y'
正解 y
最適化器
損失関数
スコア

# Keras コードとの対応



# Keras コードとの対応

```
model = Sequential()
af = 'relu'
model.add(Dense(n_hidden, input_shape=input_shape, activation=af))
model.add(Dense(n_hidden, activation=af))
model.add(Dense(n_hidden, activation=af))
model.add(Dense(n_hidden, activation=af))
model.add(Dense(n_hidden, activation=af))
model.add(Dense(n_out, activation='softmax'))
model.compile(loss = 'categorical_crossentropy',
              optimizer = RMSprop(),
              metrics = ['accuracy'])
model.fit(X_train, Y_train, epochs=10, batch_size=200)
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

活性化関数は ReLU

5 層の隠れ層のあとに
ソフトマックス計算

損失関数、最適化器等の
設定

学習

評価

# Section2 のまとめ

- 深層学習
  - 特徴抽出処理も学習の対象としているところがポイント
- 多階層のニューラルネットワークが学習可能になり ブレイク
  - 事前学習法→活性化関数の工夫
  - ドロップアウトによる汎化性能向上
- 問題に応じた構造
  - 画像：畳み込みニューラルネットワーク
  - 自然言語：リカレントニューラルネットワーク