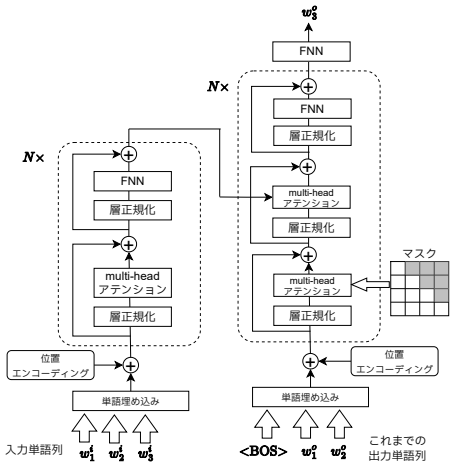
9. ニューラルネットワークの応用



- 9.1 深層学習とは
- 9.2 畳み込みニューラルネットワーク
- 9.3 リカレントニューラルネットワーク
- 9.4 Transformer

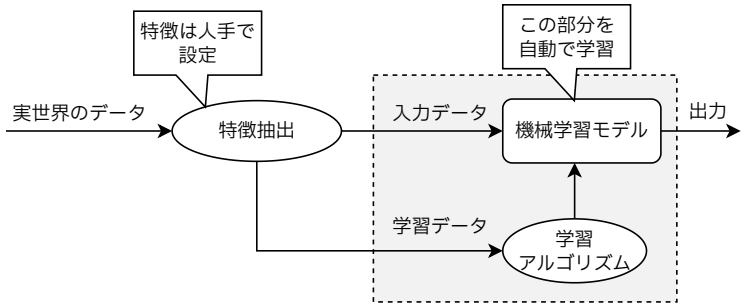
- 荒木雅弘：『Pythonではじめる機械学習』（森北出版，2025年）
- スライドとコード

9.1 深層学習とは (1/2)

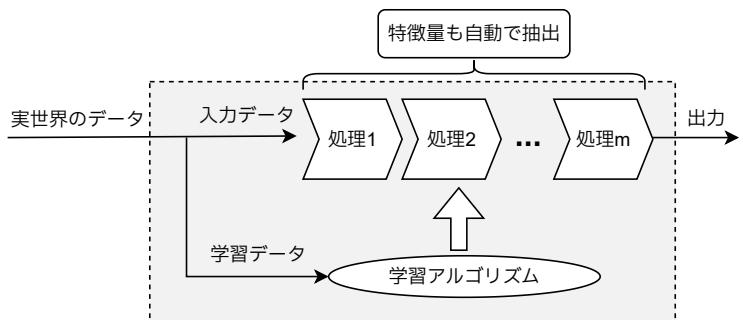
- 深層学習とは
  - 多階層のニューラルネットワークを用いた機械学習
    - 基本的な学習手段は誤差逆伝播法
  - 特徴抽出段階も学習対象とすることで大幅に性能が向上
  - 入力の種類に応じたネットワーク構造が提案された
    - 画像認識に適した畳み込みネットワーク
    - 自然言語処理に適したリカレントネットワーク
    - さまざまな応用分野を持つ Transformer アーキテクチャ
      - 近年では多くの処理が Transformer ベースの基盤モデルに集約されつつある

9.1 深層学習とは (2/2)

- 通常の機械学習モデルと深層学習モデルの位置付け



(a) これまでに説明した機械学習



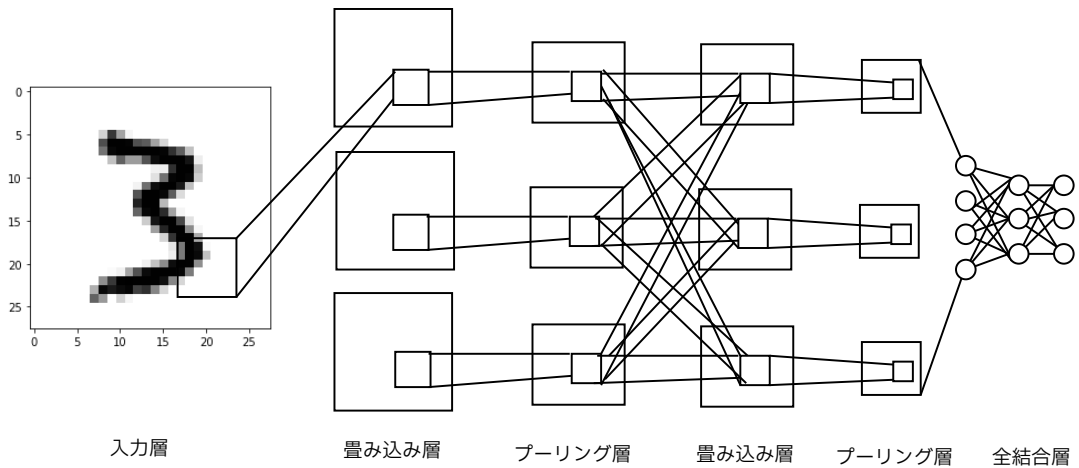
(b) 深層学習

9.2 畳み込みニューラルネットワーク (1/5)

- 畳み込みニューラルネットワーク(CNN: Convolutional Neural Network)とは
  - 画像認識や音声認識に適したニューラルネットワーク
    - 空間あるいは時間軸上に並んだ信号から特定のパターンを見つけ出す
  - 畳み込み層とプーリング層を交互に重ねて特徴抽出
    - 畳み込み層：フィルタを使って特定のパターンを見つける
    - プーリング層：位置の変動を吸収するダウンサンプリング
    - これらを交互に重ねることで，複雑な特徴を表現可能
  - 正規化層で入力値を調整することもある
  - スキップ接続を入れることもある
  - 最後は数段の FNN（ReLU+Softmax）

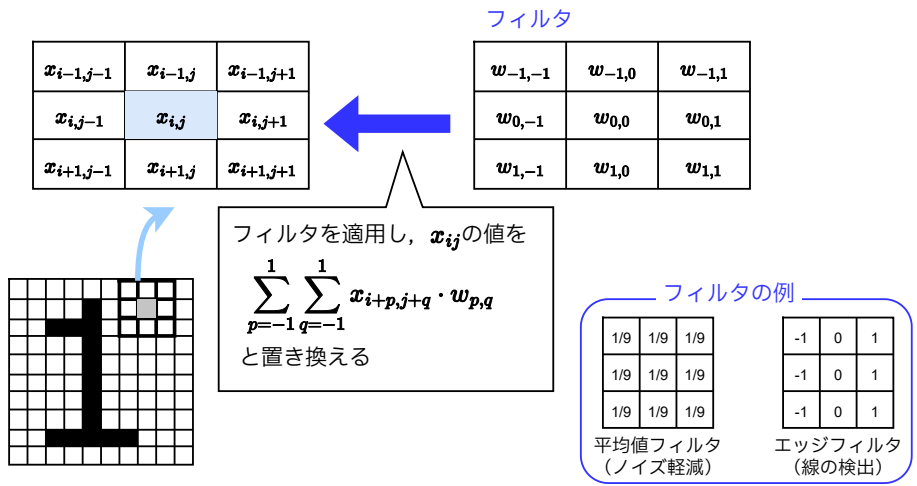
9.2 畳み込みニューラルネットワーク (2/5)

- 畳み込みネットワークの構造
  - 畳み込み層は各チャネルでそれぞれ異なるフィルタを学習
    - 重みはチャネル内では共通なので，学習対象であるパラメータの数は少ない
  - プーリング層は最大値または平均値の計算なので，その重みは学習対象外
  - 出力側の FNN は特徴を入力として識別を行う



9.2 畳み込みニューラルネットワーク (3/5)

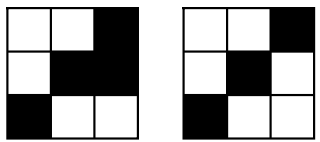
- 畳み込み層
  - 畳み込み：フィルタを一定画素（ストライド）ずつずらして画像との内積を計算



  - 畳み込み層の演算：畳み込み結果に対して活性化関数を適用
    - 結果としてフィルタに対応するパターンの出現を示す特徴マップが得られる

9.2 畳み込みニューラルネットワーク (4/5)

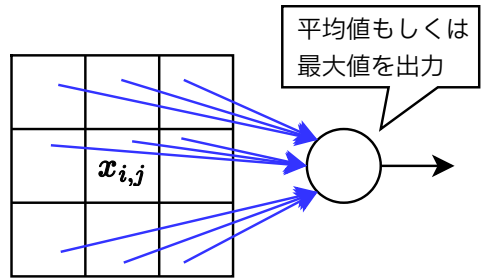
- 畳み込み演算の意味
  - 切り出した画像とフィルタパターンの類似度を求めている



  - $\boldsymbol{x}$ と $\boldsymbol{w}$ が類似しているとは，$\|\boldsymbol{x}-\boldsymbol{w}\|^2$ が小さな値となるとき
    - $\|\boldsymbol{x}-\boldsymbol{w}\|^2 = \|\boldsymbol{x}\|^2 - 2\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} + \|\boldsymbol{w}\|^2 \Rightarrow$ 内積 $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}$ が大きな値となるとき
- チャネルの意味
  - 幅 H，高さ W のカラー画像の場合，色は RGB の3チャネルで表されるので，入力は (H, W, 3) のテンソルとなる
  - 隠れ層の場合，フィルタの数がその層の出力チャネル数となる

9.2 畳み込みニューラルネットワーク (5/5)

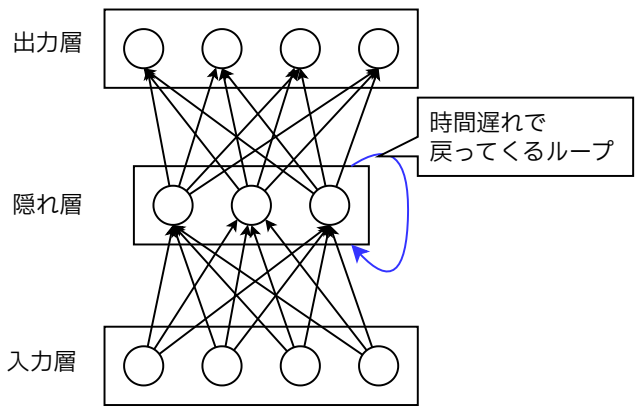
- プーリング層
  - 一定範囲の最大値あるいは平均値を計算し，畳み込み層の出力の解像度を低くする役割



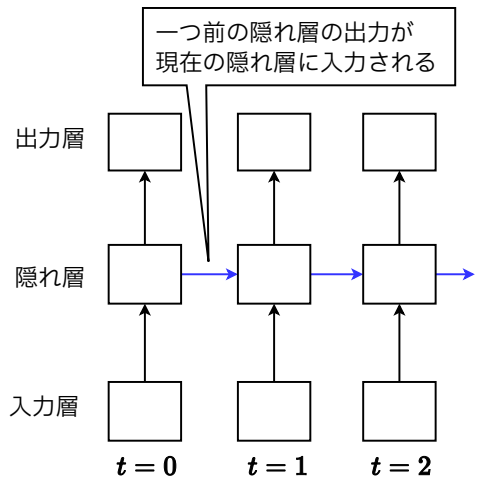
- 正規化層
  - 入力値からミニバッチ毎の平均値を引いて標準偏差で割る
  - 多階層の演算による値の大きな変動やミニバッチ毎の分布の違いを吸収
- スキップ接続
  - 学習時に誤差をそのまま伝えるので，CNN の多階層化が可能になった

9.3 リカレントニューラルネットワーク (1/10)

- リカレントニューラルネットワーク（RNN: Recurrent Neural Network）とは
  - 時系列信号の認識や自然言語処理に適したニューラルネットワーク
  - 一時刻前の隠れ層の出力を次の入力と結合して隠れ層に入力



(a) リカレントニューラルネットワーク



(b) ループを時間方向に展開

9.3 リカレントニューラルネットワーク (2/10)

- 隠れ層の計算
  - 入力を $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ としたとき，内部状態 $\boldsymbol{h}$ が時刻毎に更新されてゆくと考える

$$\boldsymbol{h}_t = f(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}_h)$$

    - $\boldsymbol{W}_h$: 隠れ層のループの重みを表した行列
    - $\boldsymbol{W}_x$: 入力層から隠れ層への重みを表した行列
    - $\boldsymbol{b}_h$: 各隠れ層のバイアス項を結合したベクトル
- 出力層の計算
  - 入力系列の識別：$\boldsymbol{h}_N$ から得られる重み付き和 $\boldsymbol{W}_y \boldsymbol{h}_N + \boldsymbol{b}_y$ の softmax 等で識別結果 $\boldsymbol{y}$ を出力
  - 入力に対するラベルの付与：各 $\boldsymbol{h}_t$ から識別と同様の手順で $\boldsymbol{y}_t$ を順次出力

9.3 リカレントニューラルネットワーク (3/10)

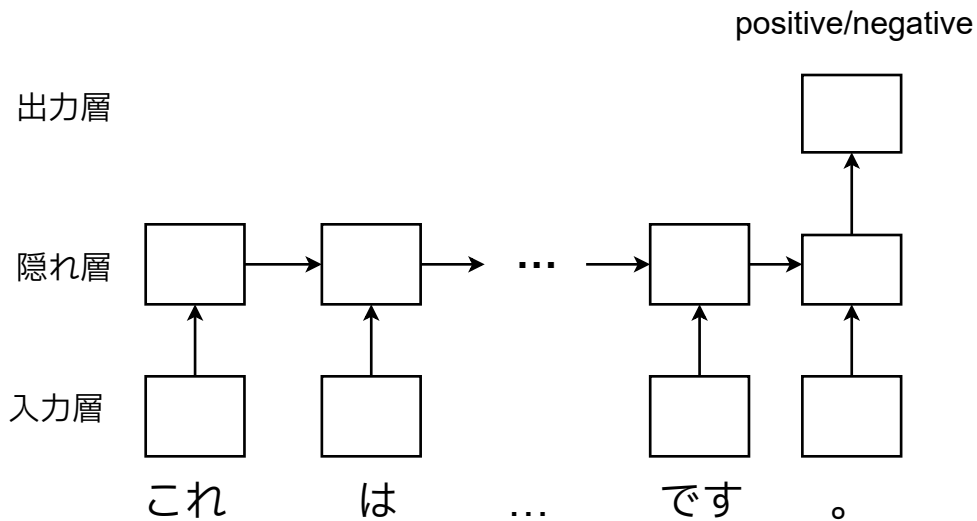
- リカレントネットワークの学習
  - 通常の誤差逆伝播法の更新式に対して，時間を遡った更新が必要
  - 時刻 t において，k 個過去に遡った更新式

$$\boldsymbol{W}_h^{'} \leftarrow \boldsymbol{W}_h - \eta \sum_t \sum_{j=0}^{k} \epsilon(t-j) \left(\prod_{i=1}^{j} \boldsymbol{W}_h^{\mathsf{T}} f^{'}(\boldsymbol{h}_{t-1}) \right) \boldsymbol{h}_{t-j-1}^{\mathsf{T}}$$

    - $\epsilon(t-j)$: 時刻 $t-j$ における誤差
    - $f'(\cdot)$: 活性化関数の微分
  - 勾配消失を避けるため，$k = 10$〜100 程度とする

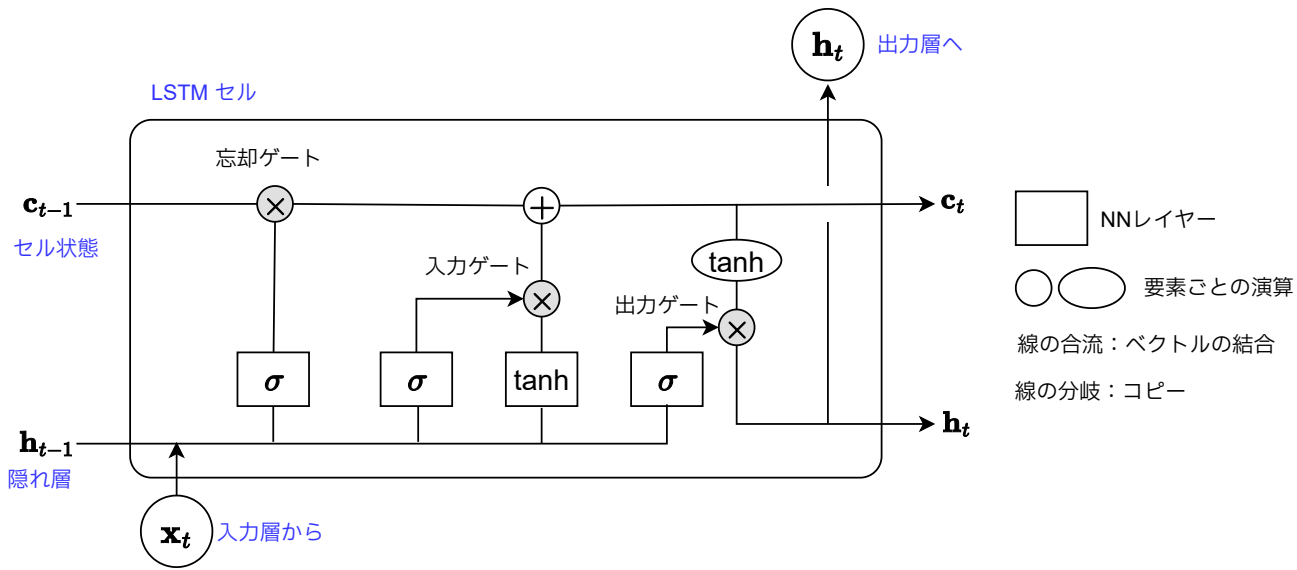
9.3 リカレントニューラルネットワーク (4/10)

- 単純なリカレントネットワークの問題点
  - 前方の情報が後方まで伝わりにくい
  - 学習時にループの重み行列が何度も掛けられ，勾配消失／爆発が生じやすい



9.3 リカレントニューラルネットワーク (5/10)

- LSTM (long short-term memory)の構造
  - 隠れ層の情報 $\boldsymbol{h}$ に加えて，ゲートによって制御されるセル状態 $\boldsymbol{c}$ を持つユニット
    - ゲート：前のセル状態，隠れ層への入力，隠れ層からの出力を選択的に通すメカニズム



9.3 リカレントニューラルネットワーク (6/10)

- LSTMのゲート
  - 忘却ゲート：セル状態のどの部分を伝えるか

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f)$$

  - 入力ゲート：入力のどの部分をセル状態の計算に用いるか

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i)$$

  - 出力ゲート：出力のどの部分を隠れ状態に組み込むか

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)$$

9.3 リカレントニューラルネットワーク (7/10)

- 隠れ層の計算

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c)$$

- セル状態の計算

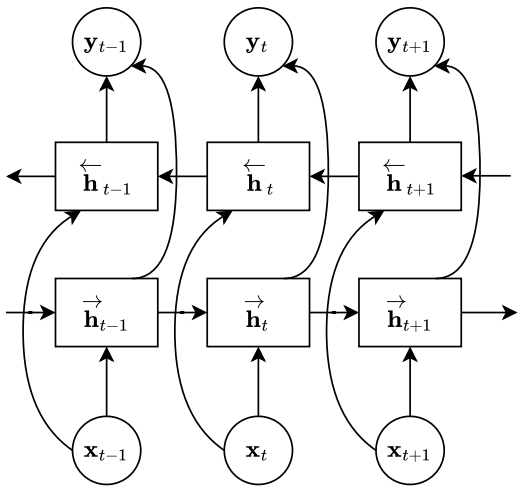
$$\boldsymbol{c}_t = \boldsymbol{f}_t * \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_t$$

  - 学習時に$\boldsymbol{W}_c$を何度も掛けることがなくなったので，勾配消失／爆発が起きにくい
- 出力の計算

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{c}_t)$$

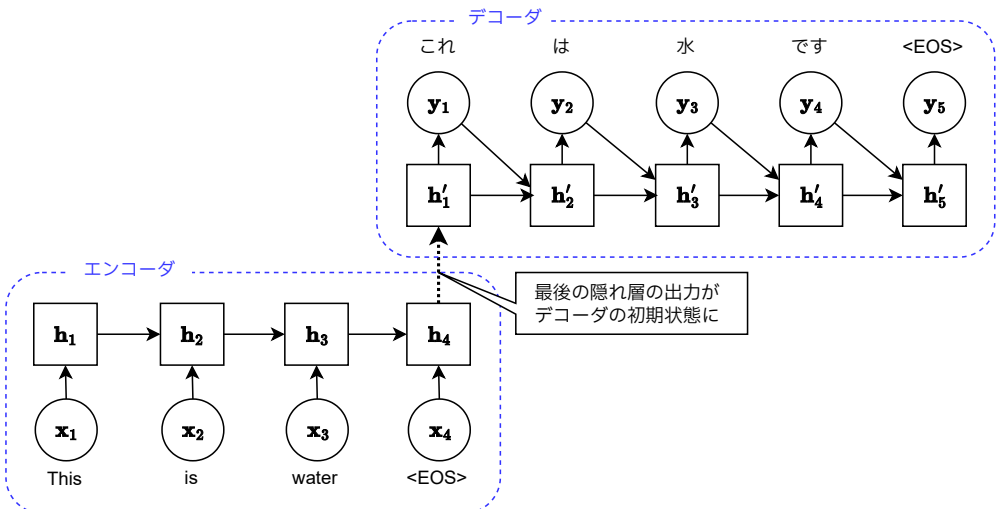
9.3 リカレントニューラルネットワーク (8/10)

- 双方向 LSTM
  - 系列全体が入力として与えられるならば，前方から順に処理する方法に限定する必要はない
  - 前方からの情報と後方からの情報を結合して出力を計算



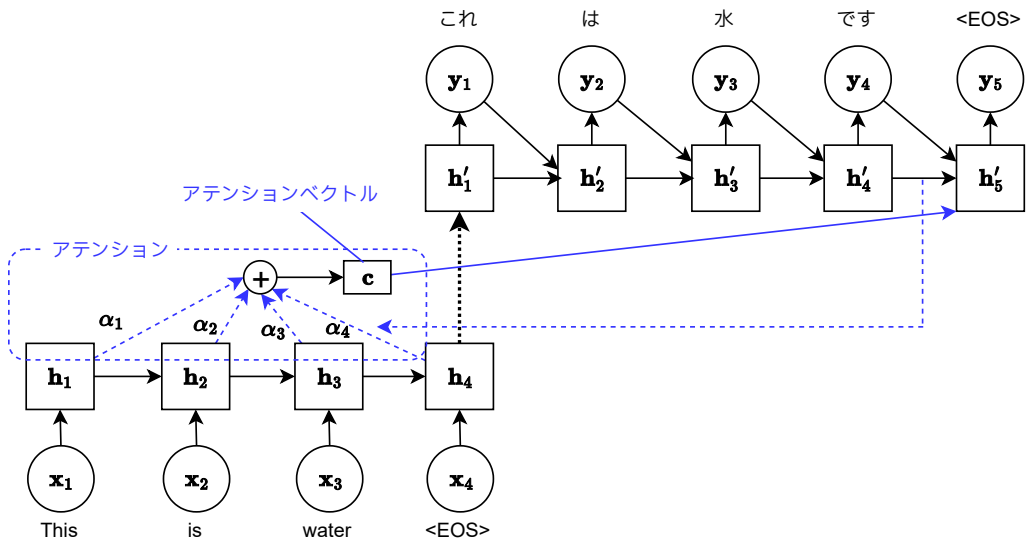
9.3 リカレントニューラルネットワーク (9/10)

- エンコーダーデコーダ構造
  - エンコーダで入力系列の内容をひとつのベクトルにまとめる
  - そのベクトルを初期値として，デコーダで出力を順次生成
  - 機械翻訳や対話システムにおける応答生成ができることが示された



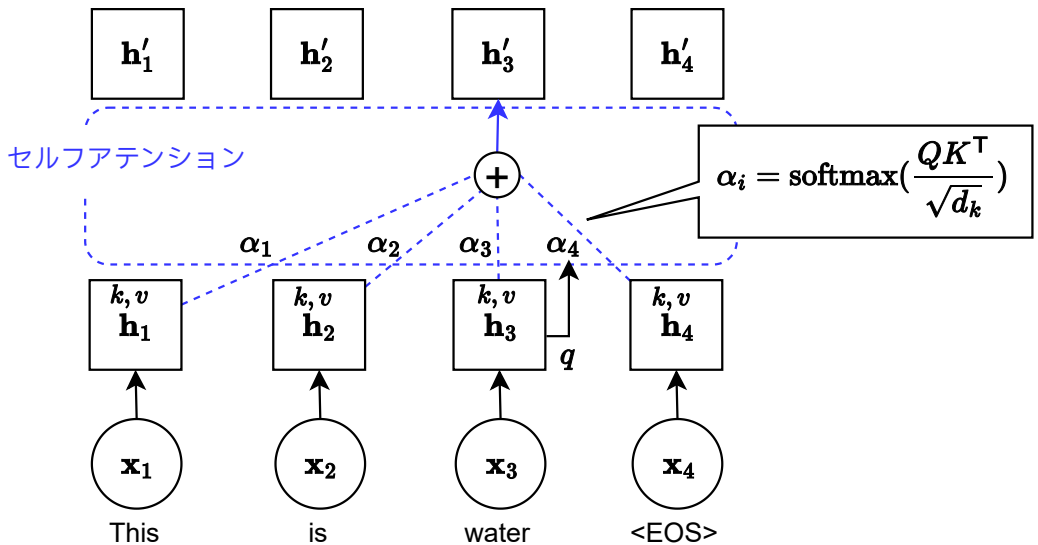
9.3 リカレントニューラルネットワーク (10/10)

- エンコーダーデコーダ構造 + アテンション
  - アテンション：エンコーダ内のすべての隠れ層出力の重み付き和
    - 入力のどの部分を見るかという情報を出力時に利用
  - 入力と出力で語順が異なる機械翻訳などに有効



9.4 Transformer (1/6)

- セルフアテンションとは
  - リカレント構造を廃止して系列全体を入力とし，各単語のベクトル表現を作成する
    - 隠れ層の出力としてベクトル表現を作るときに，入力の各単語との関係を計算



9.4 Transformer (2/6)

- セルフアテンションの計算
  - 入力 $\boldsymbol{h}$ からクエリ $\boldsymbol{q}$，キー $\boldsymbol{k}$，値 $\boldsymbol{v}$ を計算

$$\boldsymbol{q} = \boldsymbol{W}_Q \boldsymbol{h}, \quad \boldsymbol{k} = \boldsymbol{W}_K \boldsymbol{h}, \quad \boldsymbol{v} = \boldsymbol{W}_V \boldsymbol{h}$$

  - 特定の入力 $\boldsymbol{h}$ のクエリ $\boldsymbol{q}$ に対して，全入力 $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ の $\boldsymbol{k}$ との内積を計算
    - さらにこれをクエリの次元数のルート $\sqrt{d_k}$ で割って，内積の分散が1となるようにする

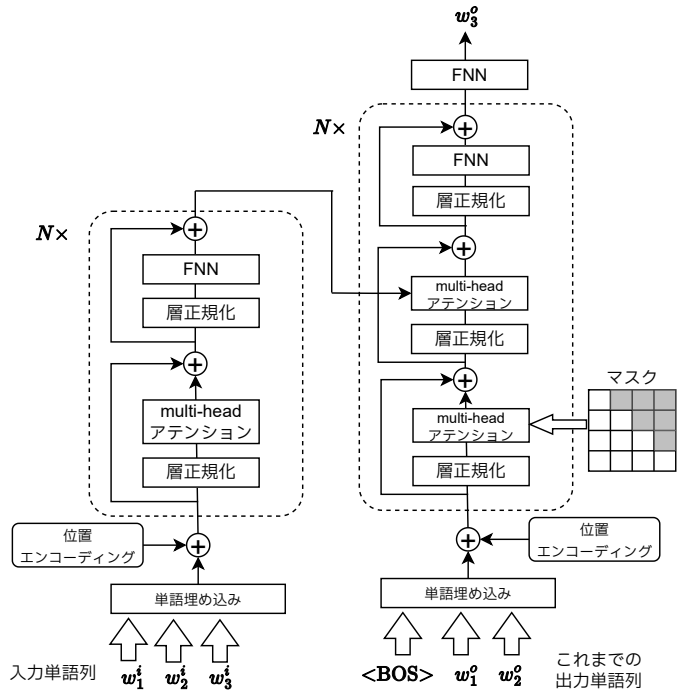
$$\alpha_i = \frac{\boldsymbol{q}^\mathsf{T} \boldsymbol{k}_i}{\sqrt{d_k}}$$

  - これらのsoftmaxを値 $\boldsymbol{v}_i$ の重みとする重み付き和によって，求めるベクトル表現を得る

$$\boldsymbol{h}' = \mathrm{softmax}(\boldsymbol{\alpha})^\mathsf{T} \boldsymbol{v}$$

9.4 Transformer (3/6)

- Transformerアーキテクチャ
  - [Vaswani et al., 2017] から層正規化の位置を現在主流のものに変更



9.4 Transformer (4/6)

- Transformer エンコーダの処理
  1. 入力の各要素を埋め込みベクトルに変換（embedding）する
  2. 位置の情報を付与（positional encoding）する
  - 位相が異なる三角関数を要素とするベクトルを埋め込みベクトルに加算する方法や，位置に応じて埋め込みベクトルを回転させる方法（RoPE）などがある
  - 位置情報を表すベクトルを学習可能なパラメータとして扱う方法もある
  3. Transformer ユニットの処理を N 段階繰り返し，入力系列に対する出力系列を得る
  - Transformer ユニットの処理
    - セルフアテンション，FNN のそれぞれの計算の前に，学習の安定化のために層正規化（各トークンごとに特徴量を正規化）を行う
    - セルフアテンション，FNN のそれぞれにスキップ接続を加える

9.4 Transformer (5/6)

- Transformer デコーダの処理
  1. 先頭記号を埋め込みベクトルに変換したものを入力とする
  2. 位置の情報を付与する
  3. デコーダ用に構造を変更した Transformer ユニットの処理を N 段階繰り返し，1トークンごとにトークンの確率分布から出力を決め，それをデコーダの入力系列に加える
     - デコーダの Transformer ユニットでは，現在処理中のトークンよりも先の情報を隠すためにマスクを用いる
     - セルフアテンションのあと，デコーダ側でクエリを，エンコーダの側でキーとバリューを計算してクロスアテンション処理を行い，その後に FNN での処理を行う

9.4 Transformer (6/6)

- Transformerを応用した基盤モデル
  - BERT : 入力側(エンコーダ)のみの構造
    - 事前学習：入力単語系列のうち，一部を [MASK] という記号に置き換え，その単語を予測する自己教師あり学習を繰り返す
    - ファインチューニング: 目的のタスクに応じてネットワーク構造を追加して再学習を行う
  - GPT : 出力側(デコーダ)のみの構造
    - 事前学習：大規模な文書データに対して，次単語を予測する問題を繰り返す
    - ファインチューニング: 目的のタスクに応じたデータで再学習を行う
  - T5 : エンコーダーデコーダ構造
    - さまざまなタスクについて，「タスク指示+入力+出力」で学習を行う（あるタスクの情報が，ほかのタスクの学習にも影響することを意図している）
    - タスクの指示と入力を与えて，出力を得る

まとめ

- 畳み込みニューラルネットワークは，信号の特徴抽出やダウンサンプリングを実現しているので，画像認識などのタスクに適する
- リカレントニューラルネットワークは，自然言語や時系列データなどの処理に適しており，LSTM やアテンション機構の導入で性能が向上した
- Transformer は，セルフアテンション機構を用いることで並列処理が可能となり，大規模モデルの学習に適している

参考文献

- 岡野原 大輔. ディープラーニングを支える技術 ――「正解」を導くメカニズム［技術基礎］. 技術評論社, 2021.
- 岡野原 大輔. ディープラーニングを支える技術〈2〉 ――ニューラルネットワーク最大の謎. 技術評論社, 2022.
- 品川 政太朗. Transformer. 電子情報通信学会誌, 107(8):826–827, 2024. (https://www.ieice-hbkb.org/files/ad_base/view_pdf.html?p=/portal/wp-content/uploads/2024/08/k107_8_826.pdf)