Rabbit Challenge 深層学習\_day3day4レポート

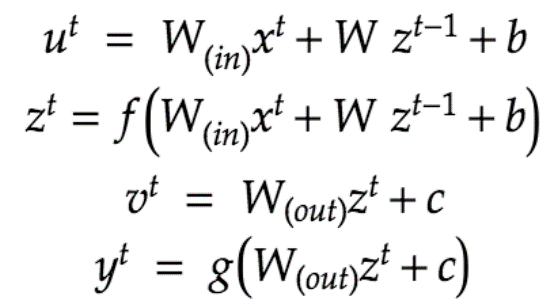
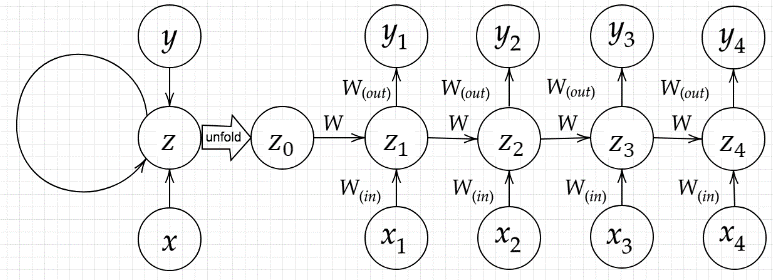
1. **深層学習day3**
   1. **Section1：再帰型ニューラルネットワークの概念**

* **要約**

〇RNN全体像

RNNとは時系列データに対応可能なニューラルネットワークである。時系列データとは、時間的順序を追って一定間隔毎に観察され、しかも相互に統計的依存関係が認められるようなデータ系列。(音声、自然言語、株価など)

RNNは以下の構造、及び数式で表現される。



f,gは活性化関数、zは中間層、uは入力層と中間層の間の活性化関数に対する入力、vは中間層と出力層の間の活性化関数に対する入力を指す。

時系列モデルを扱うには、初期状態と過去状態t-1の状態を保持し、そこから次の時間tを再帰的に求める再帰構造が必要となる。

〇BPTT(BackPropergatoinThroghTime)

RNNの誤差逆伝播の一種。

RNNのネットワーク構造、及び順伝播の数式から、BPTTの数式導出を確認。

* **実装演習結果(キャプチャーorサマリと考察)**

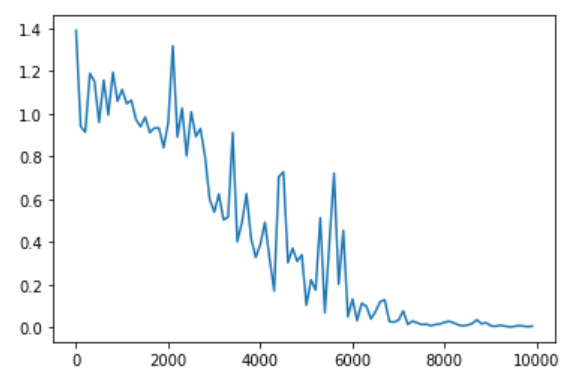
〇3\_1\_simple\_RNN.ipynb

・simple RNN

2進数の和について、各桁を時系列と捉え、桁上がりを過去状態からの入力としてRNNで学習する。

入力層サイズ2、中間層サイズ16、出力層サイズ1とする。重みの初期値にXavier、Heは利用しない。活性化関数は中間層、出力層共にsigmoid関数を利用する。

10000回の学習で、誤差は0.0067まで下がり、高い精度が出ていることを確認した。



* **確認テストなど、自身の考察結果**

〇RNNの3つの重み

入力層と中間層の間の重み(Win)、中間層と出力層の間の重み(Wout)、前回中間層と今回の中間層の間の重み(W)の3つ。

〇RNNの1つの出力を入力層x、中間層s、重みWin/Wout/Wで表現

順伝播の数式に対し、中間層の全入力u、出力層の全入力vを代入して算出。

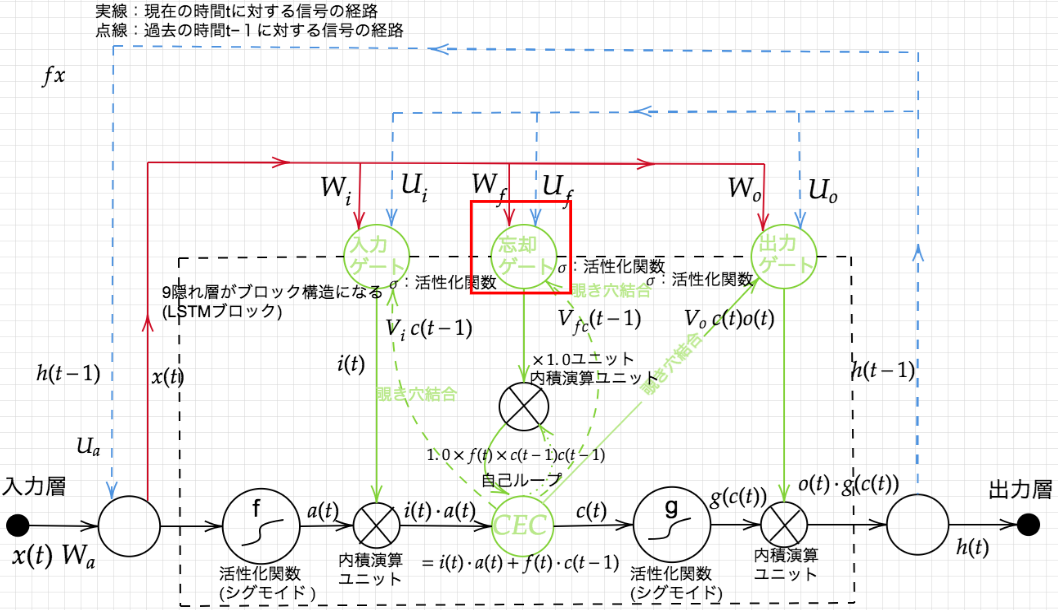
* **演習問題、参考図書、修了課題などの関連レポート**

〇構文木の活性化関数に与える引数

構文木の構造を確認。左右のノードの情報量を下げない状態で活性化関数に引数を与えることを理解した。Concatenateによる配列結合により次数が大きくなるが、次ノードの重みとの内積時に次数を下げることができることを学んだ。

* 1. **Section2：LSTM**
* **要約**

時系列を遡るほど勾配が消失するため、長い時系列の学習が困難。そこで、構造自体を変えて解決したものがLSTM。勾配爆発とは、層を逆伝播するごとに勾配が指数関数的に大きくなる事象。



〇CEC

CECは過去情報の記録装置。勾配消失及び勾配爆発の解決方法として、勾配が1であれば解決できる。一方、入力データについて、時間依存度に関係なく重みが一律であり、ニューラルネットワークの学習特性が無い。

〇入力ゲートと出力ゲート

入力ゲートは、入力層→活性化関数の出力a(t)と入力ゲートi(t)の内積結果をCECに記憶させる。すなわち、入力ゲートはCECに効果的な記録方法を学習させる。

出力ゲートは、CEC出力c(t)→活性化関数の出力g(c(t))に出力ゲートo(t)の内積結果を出力する。すなわち、出力ゲートはCEC出力の効果的な利用方法を学習させる。

両ゲートの学習は、以下の重みの計算となる。

　重みW：時間tの入力をどのように利用するかを学習

　重みU：時間t-1の出力をどのように利用するかを学習

〇忘却ゲート

CECは、過去の情報が全て保管されている。過去の情報が要らなくなった場合、削除することはできず、保管され続ける。過去の情報が要らなくなった場合、そのタイミングで情報を忘却する機能が必要。時間tでCECが記憶している情報をc(t)とすると、以下の式で表現できる。

　c(t)=i(t)・a(t)+f(t)・c(t-1)

〇覗き穴結合

CECの保存されている過去の情報を任意のタイミングで他のノードに伝播、あるいは任意のタイミングで忘却させたい。CEC自身の値は、ゲート制御に影響を与えていない。CEC自身の値に、重み行列を介して伝播可能にした構造。ただし、効果的な結果は得られなかった。

* **実装演習結果(キャプチャーorサマリと考察)**

※Sectoin3：GRUの実装演習に記載

* **確認テストなど、自身の考察結果**

〇シグモイド関数微分値の最大値

入力値が0の場合に最大値を取り、0.25。

〇LSTMによる単語予測で影響がないデータを無視するしくみ

忘却ゲートで影響がないデータを記憶しないようにする

* **演習問題、参考図書、修了課題などの関連レポート**

〇演習チャレンジ

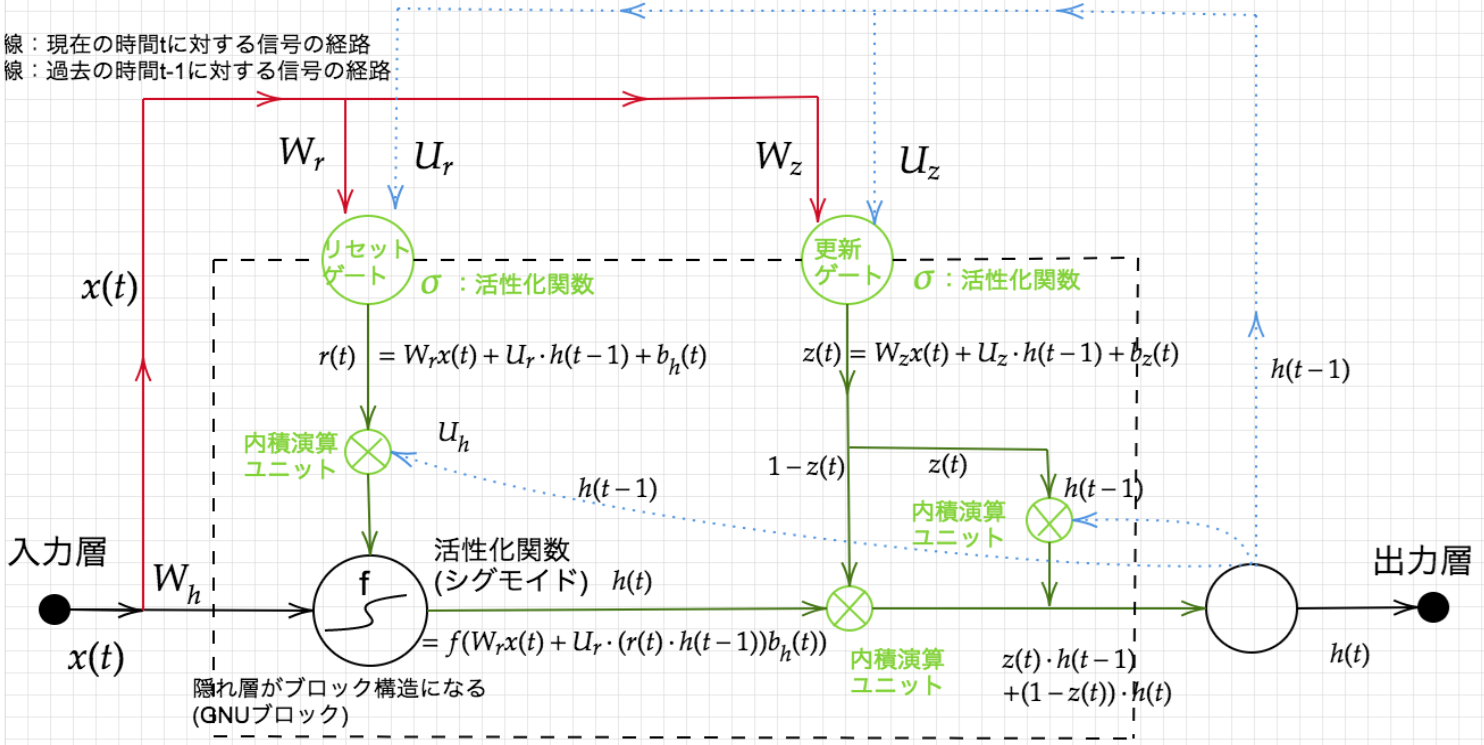
勾配爆発を防ぐために勾配のクリッピングを行う。具体的には、勾配のノルムが閾値を超えたら、勾配のノルムを閾値に正規化する。

〇演習チャレンジ

LSTMのCEC記憶情報c(t)は、入力ゲートと忘却ゲートの前回値から算出することを確認。

* 1. **Section3：GRU**
* **要約**

従来のLSTMでは、パラメータが多数存在していたため、計算負荷が大きかった。しかし、GRU(Gated Recurrent Unit)では、そのパラメータを大幅に削減し、精度は同等またはそれ以上が望める様になった構造。



〇GNUの順伝播

入力データとリセットゲートの和を活性化関数に入力し、その結果と更新ゲートの内積を行い、出力とする。

〇リセットゲート

時間tの入力x(t)、重みWrの内積と、時間t-1の出力h(t-1)、重みUrの内積にバイアスbh(t)の和で表現。

　r(t)=Wr・x(t)＋Ur・h(t-1)＋bh(t)

〇更新ゲート

時間tの入力x(t)、重みWzの内積と、時間t-1の出力h(t-1)、重みUzの内積にバイアスbz(t)の和で表現。

　z(t)=Wz・x(t)＋Uz・h(t-1)＋bz(t)

* **実装演習結果(キャプチャーorサマリと考察)**

〇Corpusクラス

ファイルを読み込み、初期化(小文字、改行をスペースに変換、特定(a-zのアルファベット等)の文字以外は空白に置換、複数スペースをスぺースに置換など)し、コーパスを作成。prepare\_dataメソッドで、訓練データとテストデータを準備する。build\_dictメソッドで、コーパスから単語の出現回数を得る。その他、情報取得やベクトルや行列への変換メソッドから成る。

〇Languageクラス

コーパスインスタンスを生成し、RNN構造の初期化、学習方法(バッチサイズ、学習率、エポック、忘却ゲートバイアス等)に関する初期化を行う。RNNの構造は、入力層：単語数、隠れ層RNNユニット数：30、出力層：単語数とする。cellはBasicRNNCell、出力層の活性化関数はsoftmax関数、損失関数はクロスエントロピー、最適化手法はadamを採用。

学習後、「some of them looks like」の次の単語を予測する。

出力結果は、辞書に登録された全単語の発生確率(softmax関数による確率)であり、「et」が予測された。

* **確認テストなど、自身の考察結果**

〇確認テスト：LSTMとCECが抱える課題

LSTMは計算量が多い。CECは勾配消失対策として勾配を１としており、学習機能がない。

〇確認テスト：LSTMとGRUの違い

LSTMは入力ゲート、出力ゲート、忘却ゲート、及びCECから構成される。

一方、GRUはリセットゲート、更新ゲートから構成され、シンプルな構成である。上記より、重みの学習量が大幅にGRUが少ない。

* **演習問題、参考図書、修了課題などの関連レポート**

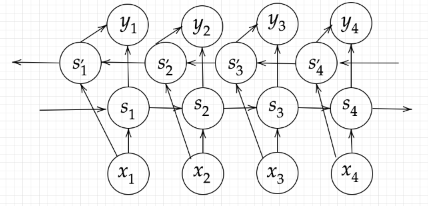
〇演習チャレンジ

GRUの更新ユニットのプログラムを確認した。

新しい中間状態は、1ステップ前の中間表現と計算された中間表現の線形和で表現される。つまり更新ゲートzを用いて、(1-z) \* h + z \* h\_barと書ける。

* 1. **Section4：双方向RNN**
* **要約**

過去の情報だけでなく、未来の情報を加味することで、精度を向上させるためのモデル。まとまった文章を一気に学習することで、過去(文章の前半)と未来(文章の後半)を利用するイメージ。実用例として、文章の推敲や、機械翻訳が挙げられる。



* **実装演習結果(キャプチャーorサマリと考察)**

無し

* **確認テストなど、自身の考察結果**

無し

* **演習問題、参考図書、修了課題などの関連レポート**

〇演習チャレンジ

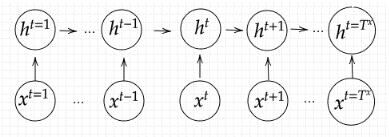
双方向RNNの出力層について、順方向/逆方向の中間層と重みの内積のプログラムについて、順方向と逆方向に伝播したときの中間層表現をあわせたものが特徴量とするためのプログラムを確認。

* 1. **Section5：Seq2Seq**
* **要約**

Encoder-Decoderモデルの一種であり、機械対話や、機械翻訳などに使用されている。

〇Encoder RNN

ユーザーがインプットしたテキストデータを単語等のトークンに区切って渡す構造。



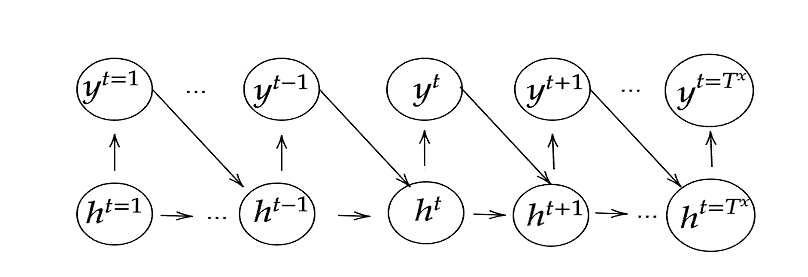
1.Taking :文章を単語等のトークン毎に分割し、トークンごとのIDに分割する。

2.Embedding :IDからそのトークンを表す分散表現ベクトルに変換(one-hotベクトルは無駄が多いため)。類似の意味を持つ単語は、ベクトル表現としても類似である必要がある。この特徴量抽出手法として、教師なし学習のMasked Language Modelが挙げられる。

3.Encoder RNN:ベクトルを順番にRNNに入力していく。vec1をRNNに入力し、hidden stateを出力。このhiddenstateと次の入力vec2をhidden stateを出力という流れを繰り返す。最後のvecを入れたときのhiddenstateをfinalstateとしてとっておく。このfinalstateがthoughtvectorと呼ばれ、入力した文の意味を表すベクトルとなる。

〇Decoder RNN

システムがアウトプットデータを、単語等のトークンごとに生成する構造。



1.Encoder RNN のfinal state (thought vector) から、各token の生成確率を出力する。final state をDecoder RNN のinitial state ととして設定し、Embedding を入力。

2.Sampling:生成確率にもとづいてtoken をランダムに選ぶ。

3.Embedding:2で選ばれたtoken をEmbedding してDecoder RNN への次の入力とします。4.Detokenize:1 -3 を繰り返し、2で得られたtoken を文字列に直す。

〇HRED

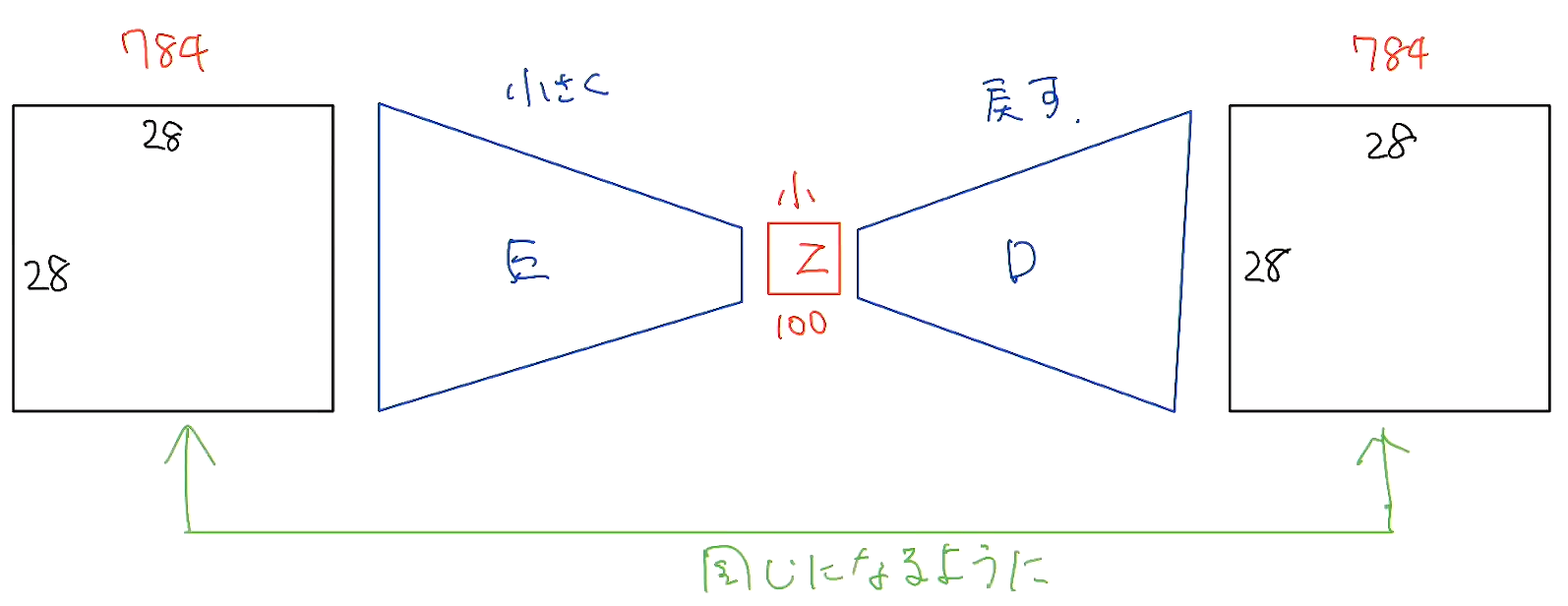
Seq2seqでは会話の文脈無視で応答がなされたが、HREDでは前の単語の流れに即して応答されるため、より人間らしい文章が生成される。HREDは、「Seq2Seq+ Context RNN」と言える。Context RNNはEncoderのまとめた各文章の系列をまとめて、これまでの会話コンテキスト全体を表すベクトルに変換する構造。ただし、HREDは確率的な多様性が字面にしかなく、会話の「流れ」のような多様性が無く、短く情報量に乏しい答えをしがちである。

〇VHRED

HREDにVAEの潜在変数の概念を追加した構造。

〇オートエンコーダー

オートエンコーダーは教師なし学習。入力データから潜在変数zに変換するニューラルネットワークをEncoder、逆に潜在変数zをインプットとして元画像を復元するニューラルネットワークをDecoder。潜在変数zを入力データより小さい構造とすることで、次元削減が可能。



通常のオートエンコーダーの場合、何かしら潜在変数zにデータを押し込めているものの、その構造がどのような状態かわからない。左記の対策として、VAE（Variational Autoencoder）はこの潜在変数zに確率分布z∼N(0,1)を仮定した構造。

* **実装演習結果(キャプチャーorサマリと考察)**

無し

* **確認テストなど、自身の考察結果**

〇Seq2Seqの妥当な説明

RNNを用いたEndoder-Decoderモデルの一種であり、機械学習翻訳などのモデルに使われる。

〇VAEの説明

事故符号化器の潜在変数に確率分布を導入したもの。

〇Seq2SeqとHRED、HREDとVHREDの違い

Seq2Seqは一問一答形式で文から文を応答する。

HREDは、Seq2Seqに文脈（コンテキスト）を覚えさせ、過去の文脈に沿った応答を行う。

HREDが短い応答しない問題に対し、VHREDはVAEを導入し、より多様な応答ができるように改良したモデル。

* **演習問題、参考図書、修了課題などの関連レポート**

〇演習チャレンジ

機械翻訳タスクにおいて入力は複数の単語から成る文であり、それぞれの単語はone-hotベクトルで表現される。Encoderにおいて、それらの単語は単語埋め込みにより特徴量に変換される。

上記処理のプログラムを確認。単語wはone-hotベクトルであり、それを単語埋め込みにより別の特徴量に変換する。これは埋め込み行列Eを用いて、E.dot(w)と書ける。

* 1. **Section6：Word2vec**
* **要約**

RNNでは、単語のような可変長の文字列をNNに与えることはできない。固定長形式で単語を表す必要がある。学習データからボキャブラリ（単語の辞書）を作成する。

ボキャブラリをone-hotベクトルを作成する。Word2Vecは、「ボキャブラリ」・「任意の単語ベクトル次元の重み行列」で表現され、大規模データの分散表現の学習が、現実的な計算速度とメモリ量で実現可能にした。

* **実装演習結果(キャプチャーorサマリと考察)**

無し

* **確認テストなど、自身の考察結果**

無し

* **演習問題、参考図書、修了課題などの関連レポート**

以下のサイトで追加学習を行った。

https://qiita.com/g-k/items/69afa87c73654af49d36

単語を固定長のベクトルで表現することを「単語の分散表現」と呼び、Word2Vecはその手法の一種である。「単語の意味は周囲の単語によって形成される」というアイデアに基づいており分布仮説と呼ばれ、Word2Vecも分布仮説に基づく。単語の分散表現を獲得する手法としては、大きく分けてカウントベースの手法と推論ベースの手法の二つがある。カウントベースの手法は周囲の単語の頻度によって単語を表現する方法で、コーパス全体の統計データから単語の分散表現を獲得する。一方で、推論ベースの手法はNNを用いて少量の学習サンプルをみながら重みを繰り返し更新する手法であり、Word2Vecは推論ベースとなる。Word2Vecは、CBOW(continuous bag-of-words)、Skip-Gramのモデルが使用されている。

* 1. **Section7：Attention Mechanism**
* **要約**

Seq2Seqは、2単語でも1000単語でも、固定次元ベクトルで表現する必要があり、長い文章への対応が難しい。そこで、文章が長くなるほどそのシーケンスの内部表現の次元も大きくなっていく、仕組みが必要となる。Attention Mechanismは、「入力と出力のどの単語が関連しているのか」の関連度を学習する仕組みである。

* **実装演習結果(キャプチャーorサマリと考察)**

無し

* **確認テストなど、自身の考察結果**

〇RNNとword2vec、seq2seqとAttentionの違い

RNNは時系列データに適したモデル。

Word2Vecは、単語の分散表現ベクトルを得るモデル

seq2seqは、1つの時系列データから別の時系列データを得るモデル

Attentoin Mechanizmは、入力と出力で重要な要素の関連度を学習するモデル

* **演習問題、参考図書、修了課題などの関連レポート**

以下のサイトで追加学習を行った。

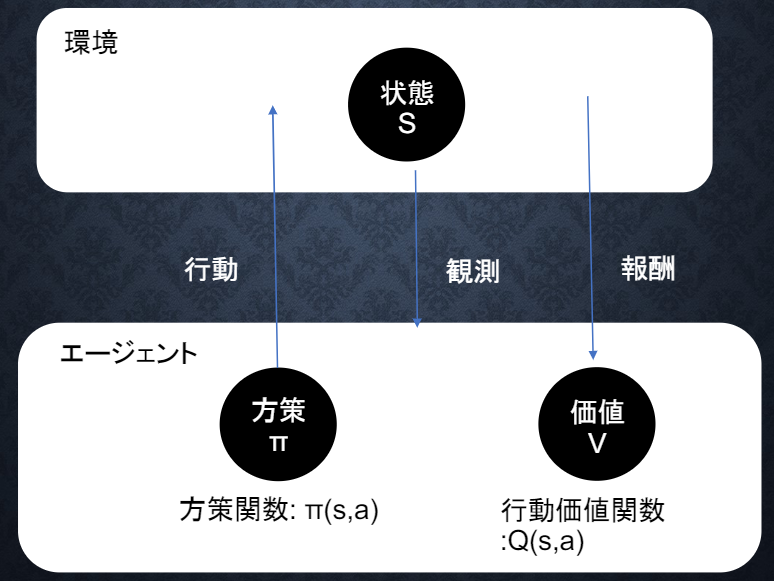
https://qiita.com/halhorn/items/614f8fe1ec7663e04bea

1. **深層学習day4**
   1. **Section1：強化学習**

* **要約**

長期的に報酬を最大化できるように環境のなかで行動を選択できるエージェントを作ることを目標とする機械学習の一分野。行動の結果として与えられる利益(報酬)をもとに、行動を決定する原理を改善していく仕組み。

不完全な知識を元に行動しながらデータを収集し、最適な行動を見つけていく。過去のデータでベストとされる行動のみを常に取り続ければ他にもっとベストな行動を見つけることはできない（探索の不足）。一方、未知の行動のみを常に取り続ければ、過去の経験が活かせない（利用の不足）。上記より、探索と利用はトレードオフとなる。



〇強化学習とその他の違い

強化学習と通常の教師あり、教師なし学習との違いは、目標が違う。

・教師なし、あり学習では、データに含まれるパターンを見つけ出すおよびそのデータから予測することが目標

・強化学習では、優れた方策を見つけることが目標

〇強化学習の歴史

強化学習の実現性は、コンピュータ性能の向上と関数近似法、Q学習を組み合わせる手法の登場による。

・Q学習：行動価値関数を、行動する毎に更新することにより学習を進める方法

・関数近似法：価値関数や方策関数を関数近似する手法のこと

〇価値関数

価値を表す関数としては、状態価値関数と行動価値関数の2種類がある

・状態価値関数：ある状態の価値に注目する

・行動価値関数：状態と価値を組み合わせた価値に注目する

Q学習では行動価値関数を用いている。

〇方策関数

方策ベースの強化学習手法において、ある状態でどのような行動を採るのかの確率を与える関数π(s,a|θ)。

方策勾配法は方策をモデル化して最適化する手法。

　θt+1=θt＋ε∇J(θ) ※θは方策関数のパラメータ

∇J(θ)の定義方法は、平均報酬、割引報酬和の2つがある。

　∇J(θ)=∇Σπ(a|s)Q(s,a)

状態sの基、方策aを行った場合の方策関数と価値関数の積を、すべての方策で和を取り、θで微分したもの

* 1. **Section2：AlphaGo**
* **要約**

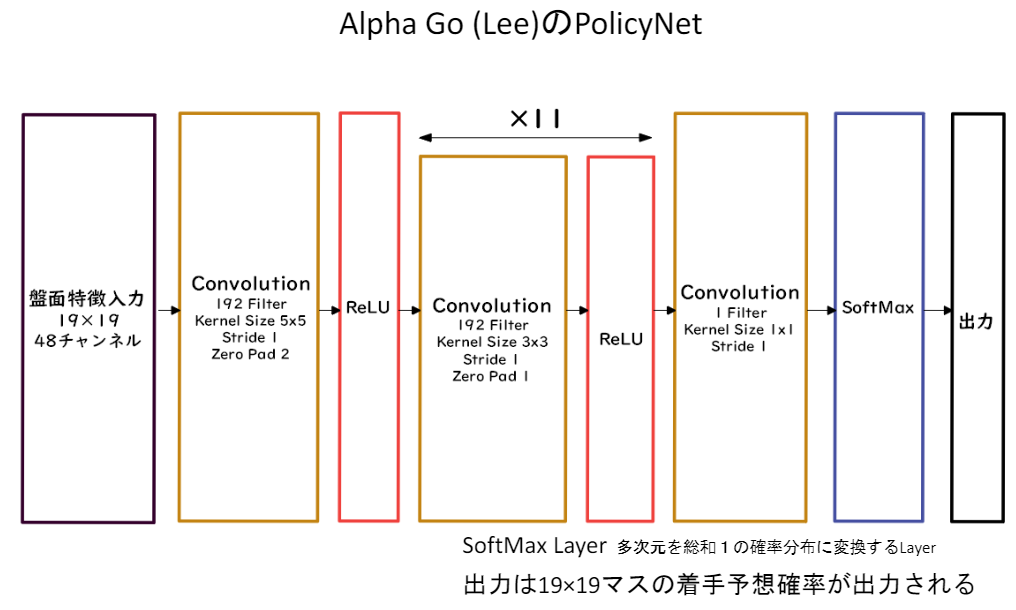
Alpha Go(Lee)、Alpha Go Zeroの2種類がある。以下はAlpha Go(Lee)の構造。

〇PolicyNetは方策関数

・入力層：19×19の48チャンネル(碁盤の状態表現で3チャンネル、着手履歴で8チャンネルなど)。

・活性化関数：SoftMax

19×19マスの着手予想が確率で出力される

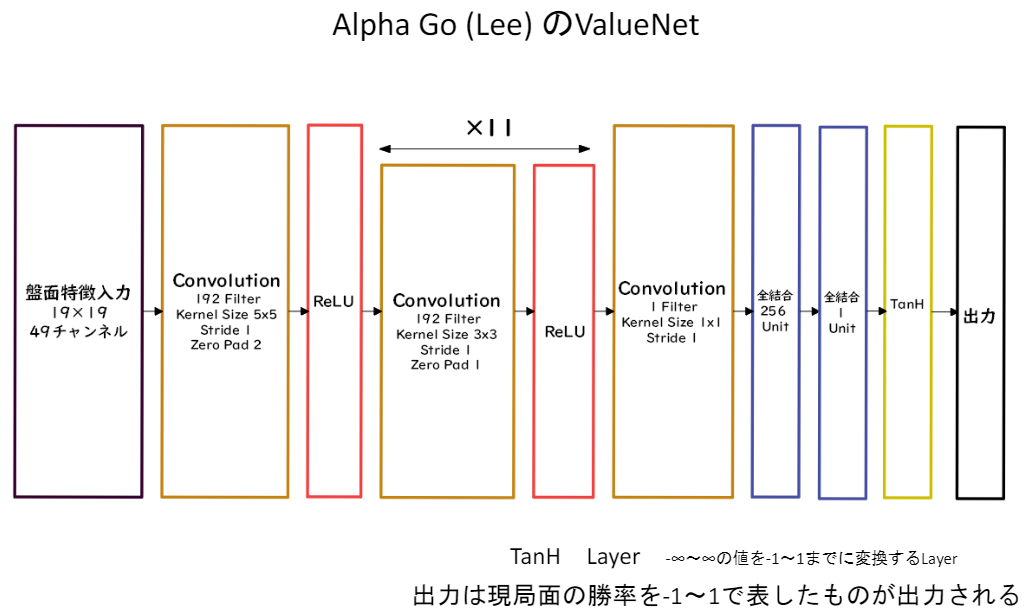


〇ValueNetは行動価値関数

・入力層：19×19の49チャンネル(PolicyNetに手番チャンネルを追加)。

・活性化関数：TanH

-∞～∞の値を、-1～1までに変換する。出力は現局面の勝率を-1～1で表したもの



〇Alpha Go学習

①教師あり学習によるRollOutPolicyとPolicyNetの学習

棋譜データから3000万局面分の教師を用意し、教師と同じ着手を予測できるよう学習。

②強化学習によるPolicyNetの学習

PolicyNetとPolicyPoolからランダムに選択されたPolicyNetと対局シミュレーションを行い、その結果を用いて方策勾配法で学習。PolicyPoolに保存されているものとの対局を使用する理由は、対局に幅を持たせて過学習を防ぐため。

③強化学習によるValueNetの学習

PolicyNetを使用して対局シミュレーションを行い、その結果の勝敗を教師として学習。

〇モンテカルロ木探索

選択、評価、バックアップ、成長という４つのステップで構成

〇Alpha Go(Lee)とAlpha Go(Zero)の違い

①教師あり学習を一切行わず、強化学習のみで作成

②特徴入力からヒューリスティックな要素を排除し、石の配置のみにした

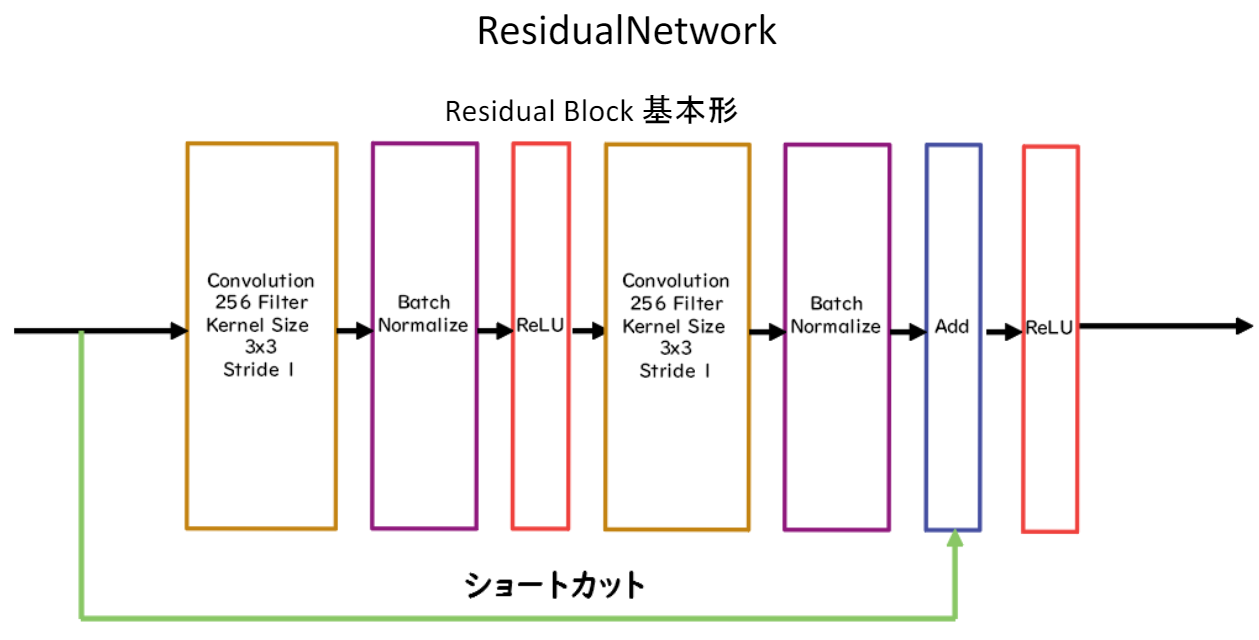
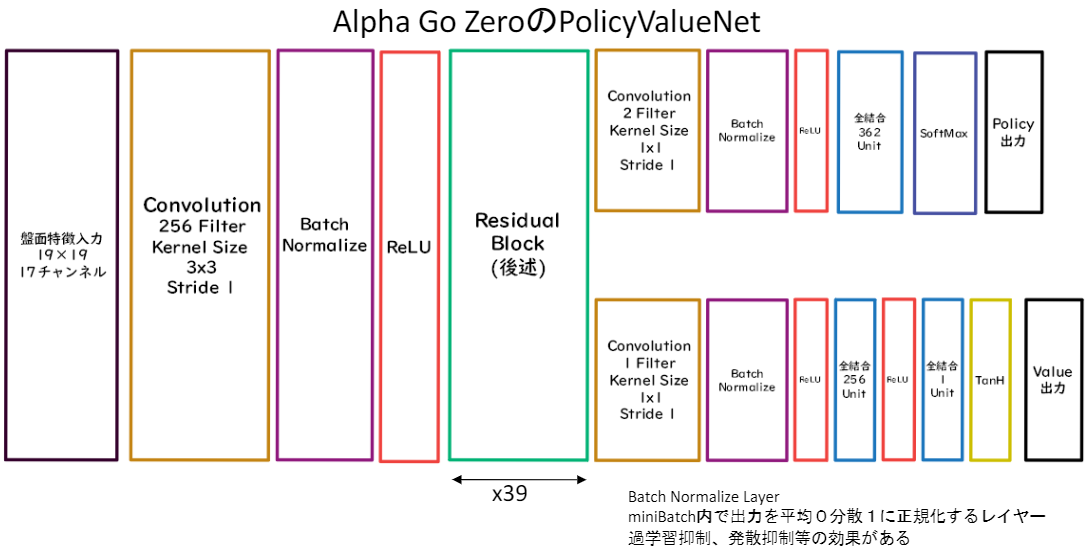
③PolicyNetとValueNetを１つのネットワークに統合した

④Residual Net（後述）を導入した

⑤モンテカルロ木探索からRollOutシミュレーションをなくした

〇Alpha Go(Zero)のPolicy Value Net

Residual Blockはネットワークにショートカット構造を追加して、勾配の爆発、消失を抑える効果を狙ったもの。基本構造はConvolution→BatchNorm→ReLU→Convolution→BatchNorm→Add→ReLUのBlockを１単位にして積み重ねる形となる。また、Resisual Networkを使うことにより層数の違うNetworkのアンサンブル効果が得られているという説もある。Batch Normalize LayerはminiBatchないで出力を0、分散を1にするレイヤーであり、過学習抑制、発散抑制等の効果がある。



* 1. **Section3：軽量化・高速化技術**
* **要約**

〇分散深層学習とは

深層学習は多くのデータを使用したり、パラメータ調整のために多くの時間を使用したりするため、高速な計算が求められる。複数の計算資源(ワーカー)を使用し、並列的にニューラルネットを構成することで、効率の良い学習を行いたい。データ並列化、モデル並列化、GPUによる高速技術は不可欠である。

〇データ並列

親モデルを各ワーカーに子モデルとしてコピー。データを分割し、各ワーカーに計算させる。データ並列化は各モデルのパラメータの合わせ方で、同期型か非同期型か決まる。同期型のパラメータ更新は、各ワーカーの計算が終わるのを待ち、全ワーカーの勾配が出たところで勾配の平均を計算し、親モデルのパラメータを更新する。非同期型のパラメータ更新は、各ワーカーはお互いの計算を待たず、各子モデルに更新を行う。学習が終わった子モデルはパラメータサーバにPushされる。新たに学習を始める時は、パラメータサーバからPopしたモデルに対して学習していく。

処理のスピードは、お互いのワーカーの計算を待たない非同期型の方が早い。非同期型は最新のモデルのパラメータを利用できないので、学習が不安定になりやすい（Stale Gradient Problem）。現在は同期型の精度が良いことが多いので、主流となっている。

〇モデル並列

親モデルを各ワーカーに分割し、それぞれのモデルを学習させる。全てのデータで学習が終わった後で、一つのモデルに復元。モデルが大きい時はモデル並列化を、データが大きい時はデータ並列化をすると良い。モデルのパラメータ数が多いほど、スピードアップの効率も向上する。

〇GPU

GPGPU (General-purpose on GPU)とは、元々の使用目的であるグラフィック以外の用途で使用されるGPUの総称。GPUは比較的低性能なコアが多数あり、簡単な並列処理が得意である。ニューラルネットの学習は単純な行列演算が多いので、高速化が可能。

GPGPUは、CUDA、OpenCLがあるが、現在の主流はCUDである。

・CUDA

GPU上で並列コンピューティングを行うためのプラットフォームであり、NVIDIA社が開発しているGPUのみで使用可能。Deep Learning用に提供されているので、使いやすい

・OpenCL

オープンな並列コンピューティングのプラットフォームであり、NVIDIA社以外の会社(Intel, AMD, ARMなど)のGPUからでも使用可能。Deep Learning用の計算に特化しているわけではない。Deep Learningフレームワーク(Tensorflow, Pytorch)内で実装されているので、使用する際は指定すれば良い。

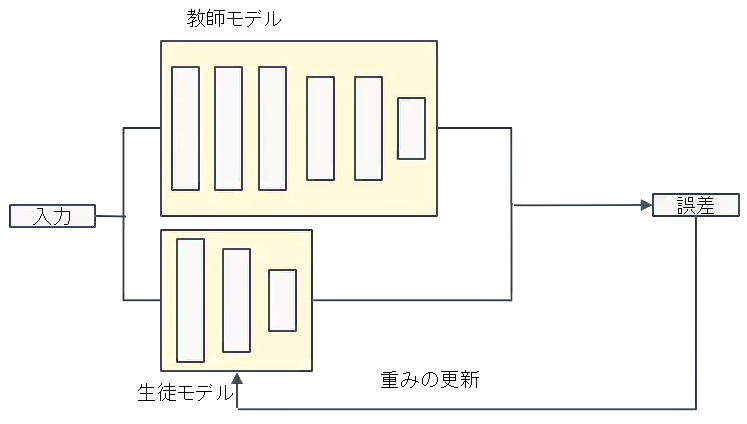
モデルの軽量化とはモデルの精度を維持しつつパラメータや演算回数を低減する手法の総称。以下の3種類が挙げられる。

〇量子化

ネットワークが大きくなると大量のパラメータが必要なり学習や推論に多くのメモリと演算処理が必要。通常のパラメータの64 bit 浮動小数点を32 bit など下位の精度に落とすことでメモリと演算処理の削減を行う。量子化の利点は計算の高速化、及び省メモリ化であるが、欠点精度の低下するデメリットもある。GPUは単精度(32bit)、倍精度(64bit)、半精度(16bit)があり、単位はFLOPを用いる。演算速度、精度のトレードオフで、現在最も利用されているのは半精度である。

〇蒸留

精度の高いモデルはニューロンが多くなり、推論に多くのメモリと演算処理が必要となる。規模の大きなモデルの知識を使い、軽量なモデルを作成することを蒸留という。蒸留は教師モデルと生徒モデルの2つで構成される。教師モデルは模倣元、生徒は教師モデルを軽量化したモデルを指す。具体的には以下のNNを利用する。



教師モデルの重みを固定し、生徒モデルの重みを更新する。誤差は教師モデルと生徒モデルのそれぞれの誤差を利用し、重みを更新する。

〇プルーニング

ネットワークが大きくなるとパラメータ数も多くなるが、すべてのニューロンの計算が精度に寄与しているわけではない。モデルの精度に寄与が少ないニューロンを削除することで、モデルの軽量化、高速化が見込まれる。ニューロンの削減手法は、重みが閾値以下の場合に削除し、再学習を行う。

* 1. **Section4：応用モデル**
* **要約**

〇MobileNets（画像認識）

Depthwise ConvolutionとPointwise Convolutionの組み合わせで軽量化を実現。Depthwise Separable Convolutionという手法を用いて計算量を削減している。通常の畳込みが空間方向とチャネル方向の計算を同時に行うのに対して、Depthwise Separable ConvolutionではそれらをDepthwise ConvolutionとPointwise Convolutionと呼ばれる演算によって個別に行う。

Depthwise Convolitionはチャネル毎に空間方向へ畳み込む。すなわち、チャネル毎にD×D×１のサイズのフィルターをそれぞれ用いて計算を行うため、その計算量はH×W×C×D×Dとなる。次にDepthwise Convolutionの出力をPointwise Convolutionによってチャネル方向に畳み込む。すなわち、出力チャネル毎に１×１×Mサイズのフィルターをそれぞれ用いて計算を行うため、その計算量はH×W×C×Mとなる。

〇DenseNet（画像認識）

初期の畳み込み、Denseブロック、変換レイヤー、判別レイヤーで構成される。

Denseブロック内には、4層で構成されており、各層の入力はそれ以前の層の出力を入力とする（チャンネル数C、k層の場合、入力チャンネル数はC×k）。また、Denseブロックの各層では、Batch正規化、活性化関数(ReLU)、畳み込みを行っている。変換レイヤーでは、畳み込みとプーリングによりもとのチャンネル数に戻す。

・DenseNetとResNetの違い

DenseBlockでは前方の各層からの出力全てが後方の層への入力として用いられる。一方、RessidualBlockでは前1層の入力のみ後方の層へ入力する。

DenseNet内で使用されるDenseBlockと呼ばれるモジュールでは成⻑率(Growth Rate)と呼ばれるハイパーパラメータが存在する。

〇BatchNormalization

レイヤー間を流れるデータの分布を、ミニバッチ単位で平均が0・分散が1になるように正規化。学習時間の短縮や初期値への依存低減、過学習の抑制など効果がある。

一方、Batch Sizeが小さい条件下では、学習が収束しないことがあり、代わりにLayer Normalizationなどの正規化手法が使われることが多い。

・Batch Norm

ミニバッチに含まれるsampleの同一チャネルが同一分布に従うよう正規化。ミニバッチのサイズを大きく取れない場合には、効果が薄くなってしまう。

・Layer Norm

それぞれのsampleの全てのpixelsが同一分布に従うよう正規化。ミニバッチの数に依存しないので、上記の問題を解消できていると考えられる。入力データのスケールや重み行列のスケールやシフトにロバストである。

・Instance Nrom

さらにchannelも同一分布に従うよう正規化。コンストラクトの正規化に寄与・画像のスタイル転送やテクスチャ合成タスクなどに利用。

〇Wavenet（音声の生成モデル）

時系列データに対して畳み込みを適用する。Dilated convolutionは、層が深くなるについて畳み込みのリンクを離し、受容野を簡単に増やすことができる。

* 1. **Section5：Transformer**
* **要約**

「Encoder-Decoder Model」→「Transfomer(Encoder-Decoder×Attention)」→「BERT」の順に学習を進める。

〇Seq2Seq

入力系列がEncode(内部状態に変換)され、内部状態からDecode(系列に変換)する。RNNと言語モデルの理解が必要。言語モデルとは、単語の並びに対して尤度、すなわち文章として自然かを確率で評価する。数式的には、同時確率を事後確率に分解して表せる。時刻t-1までの情報で、時刻tの事後確率を求めることが目標。

EncoderからDecoderに渡される内部状態が鍵。Decoderのoutput側に正解を与えれば、教師あり学習が行える。

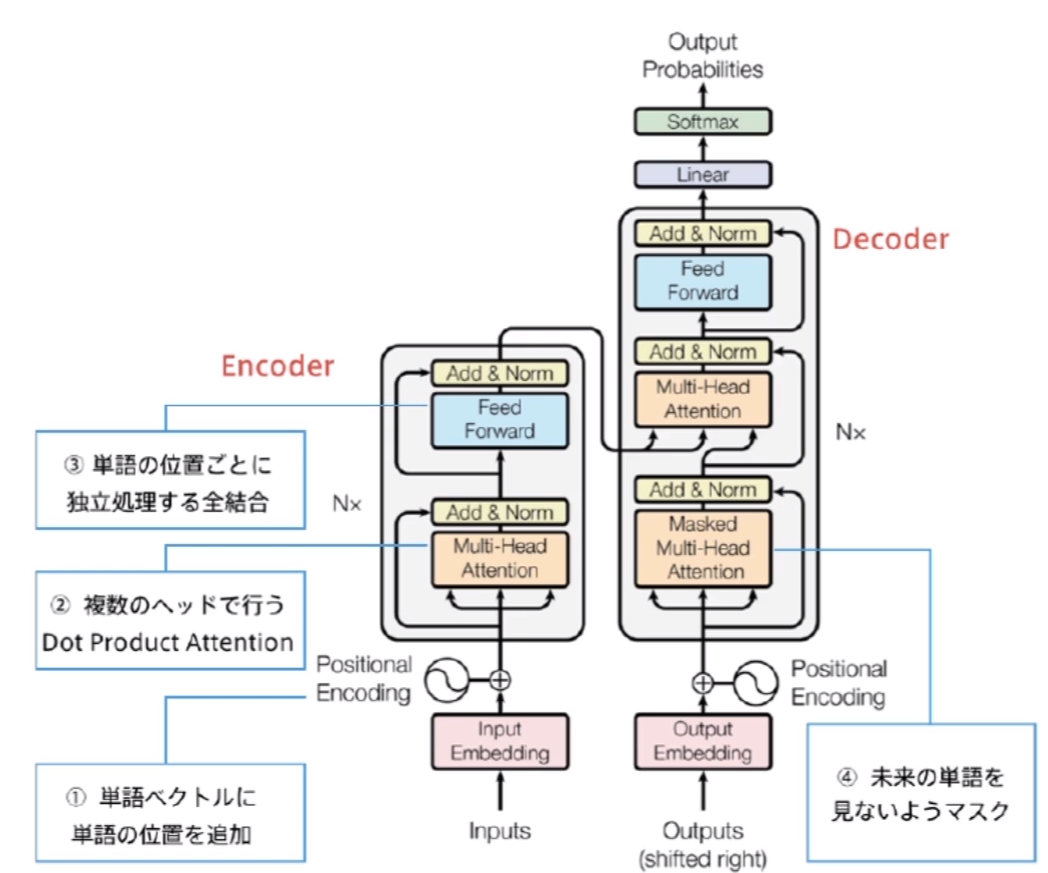
※実装確認より

Decoderの入力として、1つ前に生成したwordを次の入力にするアイデアがあるが、誤ったwordを入力すると五月雨で誤りが大きくなる。そこで、Decoderの入力に正解情報を与える(Teacher Forcing)。Teacher Forcingのみ利用すると、検証時に初めて不安定なデータが入力され、精度が悪くなる。上記より、2つをまぜたScheduled Samplingという。機械翻訳の評価指標としてBLEUがある。予め用意した複数の参照訳と機械翻訳モデルが出力した訳のn-gramのマッチに基づく指標である。

〇Transfomer

ニューラル機械翻訳の問題点として、長さに弱いことが挙げられる。そこで、Attention(注意機構)は翻訳先の各単語を選択する際に、翻訳元の文中の各単語の隠れ状態を利用する。隠れ層に対し、全て足すと1になる重みをかけることで実現。Attentionは辞書オブジェクトであり、queryに一致するkeyを索引し、対応するvalueを取り出す操作であると見なせる。

Transformerは、2017年に発表された、RNNを使わずAttentionを利用したモデルであり、当時のSOTAをはるかに少ない計算量で実現。



Encoderは自己注意機構により文脈を考慮して各単語をエンコードしている。

RNNを利用しないため単語の位置情報を個別に処理し、Attentionが利用されている。AttentionにはSource Target AttentionとSelf-Attentionの2種類がある。Self-Attentionでは、入力を全て同じにして学習的に注意箇所を決めていく。CNNはフィルタサイズが決まっており、畳み込みによりそのフィルタの抽象表現を得る。一方、Self-Attentionは文全体に対する抽象表現を取得しており、入力を全て同じにして学習的に注意箇所を決めていく。Position-Wise Feed-Forward Networksは、位置情報を保持したまま順伝播させる。Multi-Head attentionは、重みの異なる8個のScaled dot product attention(全単語に関するAttentionをまとめて計算)を使用し、それらを結合している。Decoderの特徴として、Source Target AttentionとSelf-Attentionの両方が利用されている。RNNを用いないので単語列の語順情報を追加する必要がある(Position Encoding)。

* 1. **Section6：物体検知・セグメンテーション**
* **要約**

〇物体認識タスクは、以下に分けられる。

・Classification(分類)：(画像に対し)クラスラベル

・Object Detection(物体検知)：Bounding Box、Confideceも出力されることが多い

・Semantic Segmentation(意味領域分割)：(各ピクセルに対し)クラスラベル

・Instance Segmentation(個体領域分割)：(各ピクセルに対し)クラスラベル

〇代表的データセットは以下の通り。

・VOC12(Visual Object Classes)：Instance Annotation有、クラス20数、データ11,540件、Box/画像2.4

・ILSVRC17(ImageNet Scale Visual Recognition Challenge)：クラス数200、データ476,668件、Box/画像1.1 ※ImageNetのサブセット

・MS COCO18(Common Object in Context)：Instance Annotation有、クラス数80、データ123,278、Box/画像7.3

・OICOD18(Open Images Challenge Object Detection)：Instance Annotation有、クラス数500、データ1,743,042、Box/画像7.0　※Open Image V4のサブセット

※Open Imageは画像サイズが一様でない

※Box/画像が小さいと日常とかけ離れた画像となる

目的に応じたBox/画像、クラス数(クラス分類の粒度)の選択をすること。

〇物体検出において、クラスラベルだけでなく、物体位置の予測精度も評価する。Ground-Truth BBとPredicted BBの重なり具合を評価する。

　IoU(Intersection over Union)=Area of Overlap/Area of Union

Average Precision(AP)

confidenceの値を変化させ、PrecisionとRecallの値を取得し、PR曲線を描き、AP(PR曲線の下側の面積)を取得する。APはクラス毎に算出する。全クラスの平均をとったものをmean Average Precision(mAP)。

応用上の要請から検出精度に加え、検出速度も問題となる。検出速度の指標として、FPS(Flames per Second)、interface time(1フレームの推論にかかる時間)が挙げられる。

〇深層学習以降の物体検知

・ベースネットワークの経緯

　2014年：VGGNet・GoogLeNet、2015年：ResNet、2016年：DenseNet、2017年：MobileNet、2018年：AmoebaNet

・物体検知の経緯

2段階検出器は、候補領域の検出とクラス推定を別々に行い、相対的に精度が高い。ただし、計算量が多く推論も遅い。RCNN、SPPNet、RFCN,FPN、Mask RCNNが挙げられる。1段階検出器は、候補領域の検出とクラス推定を同時に行い、相対的に精度が低い。ただし、計算量が少なく推論も早い。DetectorNet、YOLO、CornerNetが挙げられる。

〇SSD(Single Shot Detector)のアルゴリズムは以下の通り。

　１．Default BOXを用意(ランダムなBB)する。

　２．Default BOXを変形し、confidenceを出力

多数のDefault Boxを用意したことで発生する問題への対処

　・Non-Maximum Suppression：クラスが少ない場合にBBが冗長となる。IoUを計算し、IoUが高く物体が重複するBBは、conficenceが高いものだけを残す。IoUが小さい場合は別のものを指している可能性がある。

　・Hard Negative Mining：背景と物体の比率が、背景の方が多くなり不均衡となる事象が発生する。そこで、背景のBBを削除する。損失関数は、confidenceと物体検知の誤差の和で表現される。

〇Semantic Segmentationの概略

Semantic Segmentation(意味領域分割)は、入力画像の各ピクセルに対しクラスラベルを付与する。

畳み込み学習とプーリングにより、元画像から解像度が落ちてしまい、これを元の画像の解像度に戻す必要がある。これをUp-samplingという。そもそも画像認識に畳み込みが必要な理由は、受容野にある程度の大きさが必要であるため。典型的なアプローチとして、深い畳み込み層、プーリングが挙げられる。ただし、深い畳み込み層には演算量とメモリの問題がある。

〇FCN(Fully Convolutional Network)の基本的なアイデア

VGGの全結合層を全畳み込み層に変更しており、出力がHeatmapとなる。ここからさらにUp-samplingする。

Up-samplingの手法として、Deconvolution/Transposed convolutoinがある。

kernel size、padding、strideを指定し、以下の手続きを行う。

１．特徴マップのpixel間隔をstrideだけ開ける

２．特徴マップのまわりに(kernel size -1) - paddingだけの余白を作る

３．畳み込み演算を行う

Up-samplingだけでは正確に復元できない。そのため、入力層からsamplingの過程の特徴マップと和をとり、それを入力データとして用いることで、精度を向上させている。（Skip-connection）その他のアルゴリズムとしてU-Netが挙げられ、同様の考え方でアプローチしている。ただし、Skip-connectionは、チャネル方向に結合している。上記までは、Deconvolution(⇔Convolution)であり、以降はUnpooling(⇔Pooling)である。Max Pooling処理時に最大値となったセル情報(switch variables)を保持しておき、Unpooling時に再生のためにこれを利用する。Convolutionの段階で受容野を広げる工夫として、Dilated Convolutionも挙げられる。