山田　克拡 提出レポート　深層学習前半［Day3とDay4］（講義動画と実装演習）

**深層学習: 前半Day3**

**各章の要点**

**Section1：再帰型ニューラルネットワークの概念**■再帰型ニューラルネットワークRNNとは、音声データやテキストデータなどの時系列データに対応したニューラルネットワークである。

■BPTTとは、系列データを使用したRNNにおいて損失関数が最小になるようにパラメータを調整する方法で、誤差逆伝播法の一種である。

**Section2：LSTM**

■RNNにおいて、時系列をさかのぼるほど、勾配消失していく問題があったが、

LSTMは勾配消失しにくい構造になっている。

■LSTMには、入力ゲート/出力ゲート/忘却ゲートがある。

従来の方法だと、過去の情報がすべて保存されていたが、忘却ゲートを設定し、過去のセルの値に０～１を掛けることにより、過去の情報を一部or全部忘却できる。

Section3：GRU

■LSTMはパラメータ数が多く、計算負荷がかかるという課題があった。その課題を解決する方法として、GRUがある。GRUでは、パラメータを大幅に削減し、必ずではないが精度は同等以上になる。LSTMの方が精度が高い場合あるので、必ずGRUを使えばよいということではない。

Section4：双方向RNN

■過去の情報だけではなく、未来の情報を加味して、RNNの予測精度を向上させる手法。自然言語処理でよく使用される。

Section5：Seq2Seq

■seq2seqとは、Encoder-Decoderの一種であり、機械翻訳に使われている。

　一問一答式に強い。

Section6：Word2vec

■RNNはボキャブラリー×ボキャブラリーの重み行列を作るのに対して、

Word2vecはボキャブラリー×任意の単語ベクトルの重み行列を作ることで、

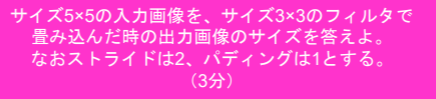
現実的な計算速度、メモリー使用を実現した。

Section7：Attention Mechanism

■Seq2seqは固定次元ベクトルしか学習できなかったが、Seq2seq+Attentionの組み合わせにより重要度/関連度を用いることにより長い文章でも学習できるようになった。

確認テストの考察

Day3 【ｐ11】

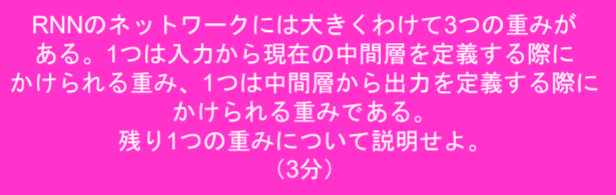


■課題

出力画像は３×３である。

（５＋1\*2-3）/2+1=3

【ｐ２３】



■課題

残り１つの重みは現在の中間層から次の中間層を定義する際にかけられる重みである。

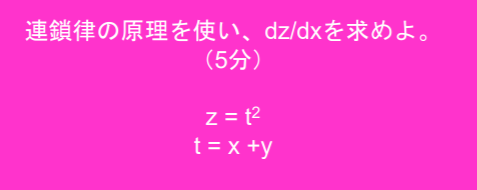
【ｐ26】



■課題

答えは（２）である。

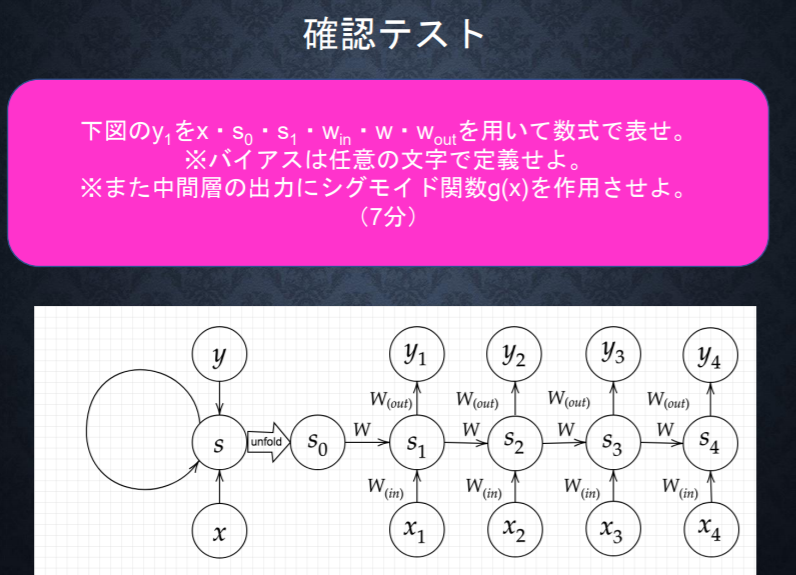
【ｐ３７】



■課題

ｄｚ/ｄｘ＝（ｄｚ/ｄｔ）（ｄｔ/ｄｘ）＝２ｔ＝２（ｘ＋ｙ）

【ｐ４６】



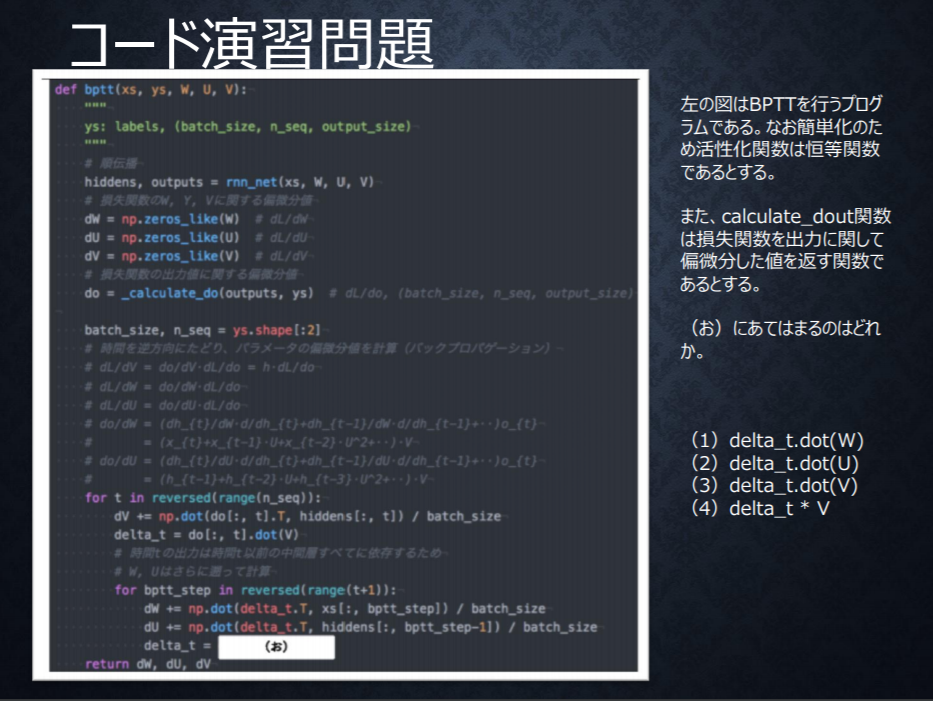
■課題

U1= W(in)\* x1+W\*S0+b

Z1=S1=sigmoid (U1) (中間層の活性化関数はsigmoid関数)

Y1=sigmoid(W(out)\*Z1+b’)　（出力層の活性化関数はsigmoid関数）

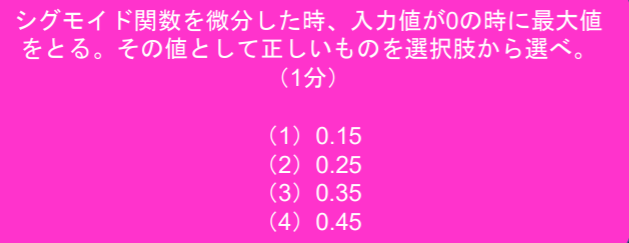
【ｐ５４】



■課題

正解は（２）である。

【ｐ６３】



■課題

（２）が正解である。

（１－０．５）＊０．５＝０．２５

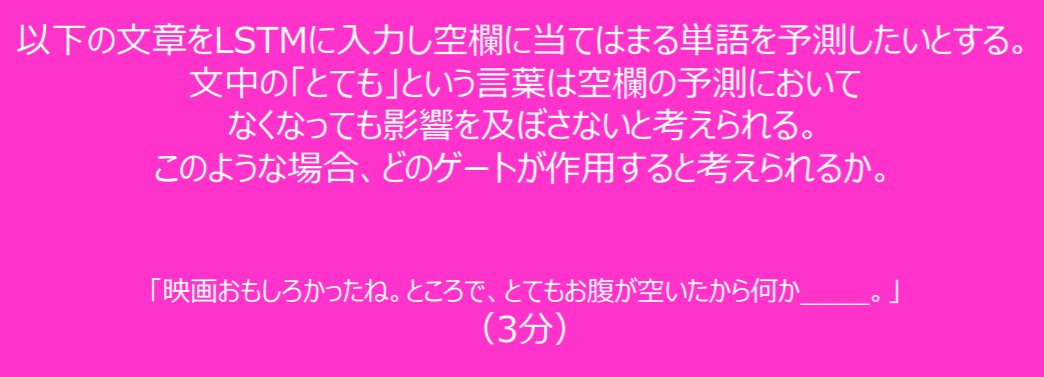
【ｐ６５】



■課題

（１）が正解である。

【ｐ７９】



■課題

忘却ゲートが作用していると考えられる。

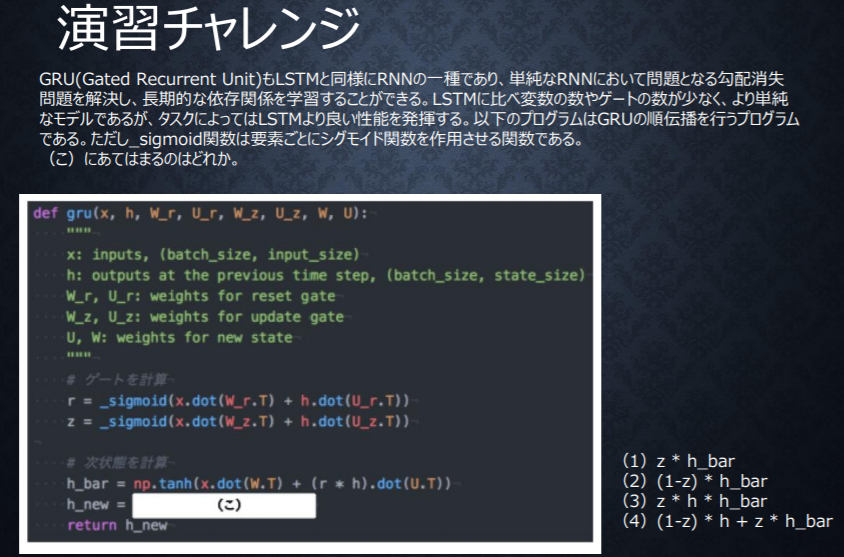
【ｐ８０】



■課題

（３）が正解である。

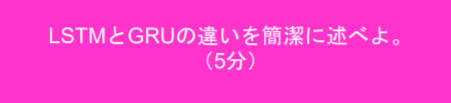
【ｐ９１】



■課題

（４）が正解である。

【ｐ９３】



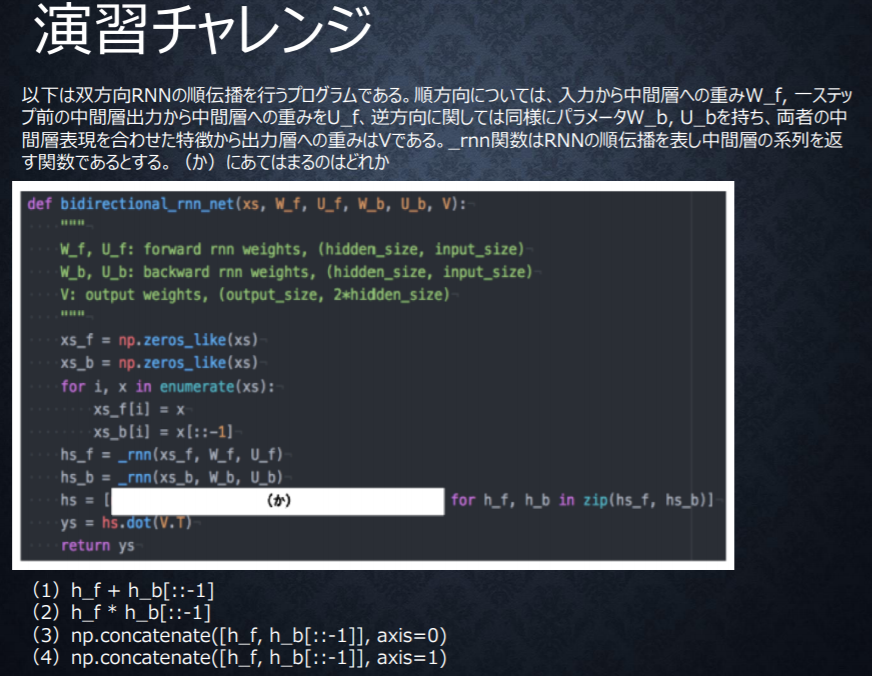
■課題

LSTMはパラメータが多く、計算負荷が大きい。

GRUはパラメータを大幅に削減したため、計算負荷が減ったが精度は同等以上である。

CECは勾配が常に１で、重みが画一化してしまい、学習自体が行えず最適化されない。

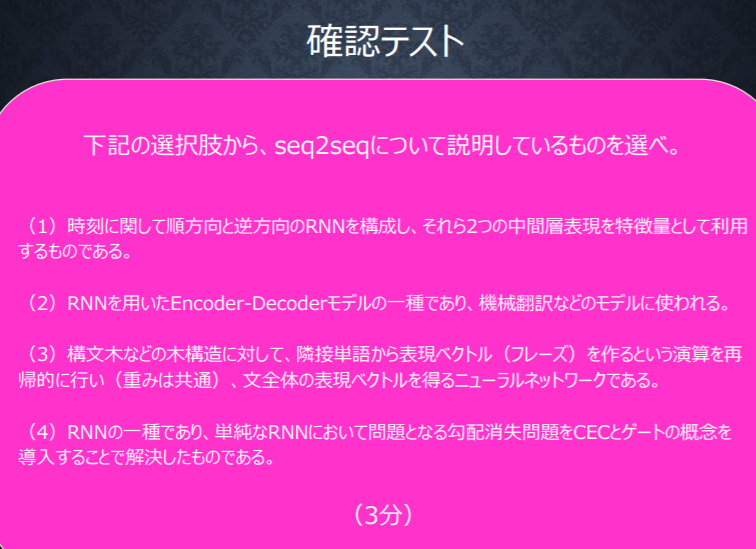
【ｐ９６】



■課題

（４）が正解である。

【ｐ１１０】



■課題

正解は（２）である。

ちなみに、（１）はBidirectionalRNN　（４）はLSTMの説明である。

【ｐ１１１】



■課題

（１）が正解である。

【ｐ１２０】

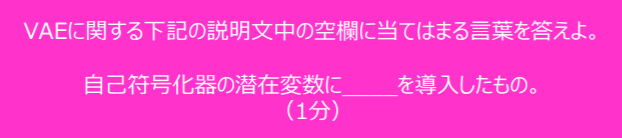


■課題

Seq2seqは一問一答しかできないのに対し、HREDは文脈に即して回答ができる。

HREDは会話の文脈に即すが毎回同じ回答だが、VHREDは毎回異なる回答が返ってくる。

【ｐ１２９】



■課題

自己符号化器の潜在変数ｚに確率分布（平均０、分散１の標準正規分布）を導入したもの

【ｐ１３８】



■RNNはボキャブラリー×ボキャブラリーの重み行列を作るのに対して、

Word2vecはボキャブラリー×任意の単語ベクトルの重み行列を作る。

Seq2seqは固定次元ベクトルしか学習できなかったが、Seq2seq+Attentionは重要度/関連度を用いることにより長い文章でも学習できるようになった。

**演習考察**

[[**DN42\_ソース演習①**](https://vimeo.com/showcase/5628412/video/304723061)](https://vimeo.com/showcase/5575638/video/302234447)**＆**[**DN43\_ソース演習②**](https://vimeo.com/showcase/5628412/video/304972338)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/3_1_simple_RNN_ensyu.pdf>

■考察

・重みの初期値をXavierにすると学習が早くなる。Heにするとさらに学習が早くなる。

・weight\_init\_stdを大きくすると、勾配爆発が起こり収束しなくなる。

・learnig\_rateを大きくすると、収束しなくなる。

・中間層のノード数を減らすと学習が進まない。

・中間層の活性化関数をreluからtanhに変更すると、勾配爆発もしくは勾配消失が起こらなくなり、学習が進むようになった。simpleRNNにおいてtanhは有効である。

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/3_3_predict_sin-ensyu.pdf>

■考察

・simpleRNNの反復回数を５００から１００に変更すると、回数が減るため、予測精度が悪くなる。

・simpleRNNの反復回数を５００から３０００に変更しても、学習がうまく進まず、予測精度は変わらない。理由は時系列データの数が２と少ないからである。時系列データを２から５に変更してやると、反復回数と時系列データ数の増加により、予測精度が上がる。

**深層学習: 前半Day4**

**各章の要点**

Section1：Tensorflowの実装演習

■GoogleがリリースしているTensorflowを使って、ディープラーニングの実装を行うことが多い。Tensorflowはsessionをrunしないと実行されない。

■Placeholderは値などを格納する入れ物で、後から値を代入することができる。

Section2：強化学習

■強化学習とは行動を決定する原理を改善していき、長期的に報酬を最大化できるようにする仕組みである。

■注意点として、過去のデータの中でベストな行動だけとっていると、さらに良い行動を探索できない。

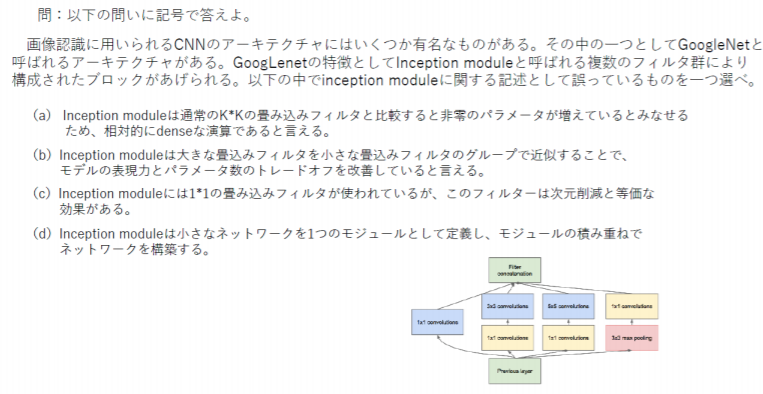
■方策は方策関数、行動は状態価値関数/行動価値関数で表される。

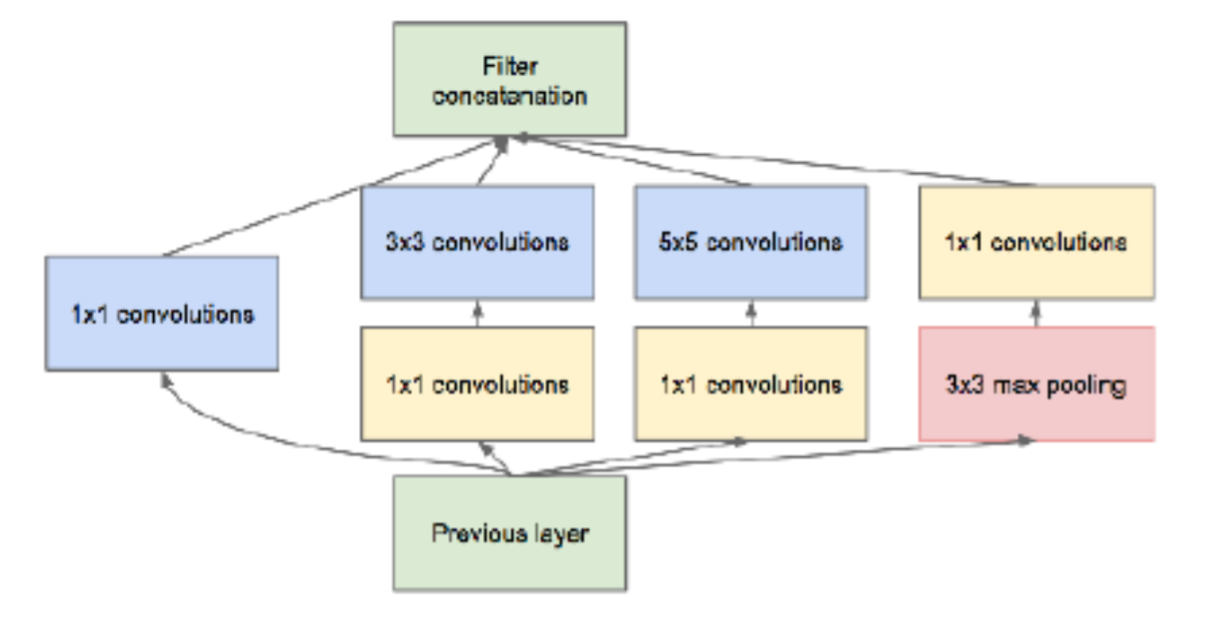
■Q学習とは行動のたびに行動価値関数を更新する学習方法である。

■方策関数とは、ある状態で各行動の確率を与える関数である。

確認テストの考察

【ｐ８】





■考察

(a)の記述が間違っている。（DenseではなくSparse）

１．googLenetは、inceptionモジュールの積み重ねで構成されたネットワークである。

２．Inceptionモジュール

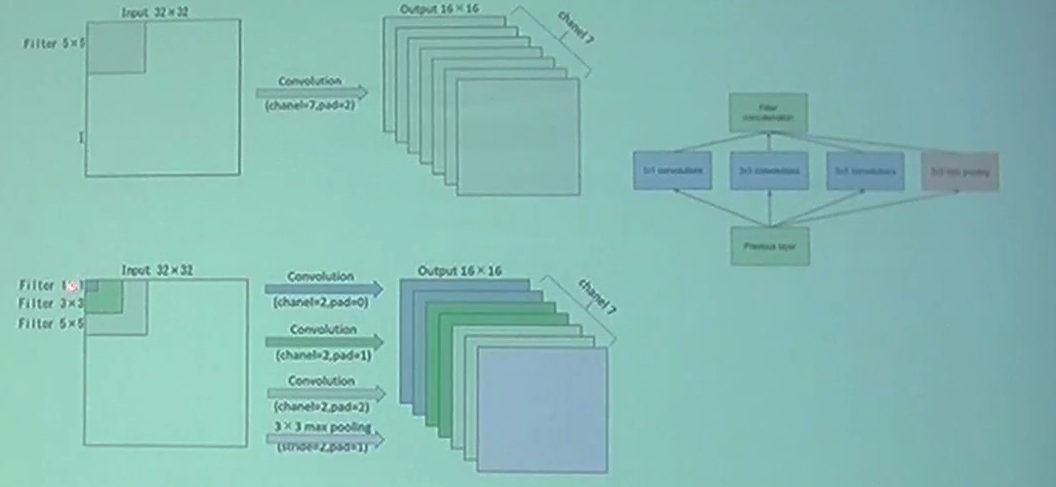
　　上図で示しているように小さいネットワークを一つのモジュールとして定義する。

　　以下、2段階にて詳細説明を行う。

1. 上図の黄色部分を除いて、伝統的なCNNとの比較

上記図の黄色部分を除くと、下記図になる。

下記図の例のように、同じ出力データサイズで、フィルタ１×１とフィルタ3×３、フィルタ5×５と複数のフィルタがあるため、伝統的なCNNより表現力が改善された。

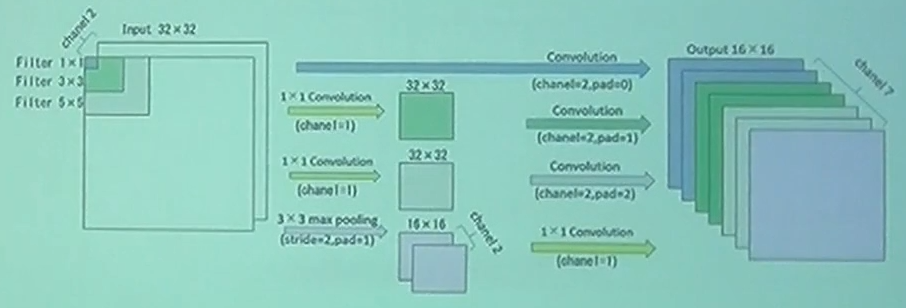


通常

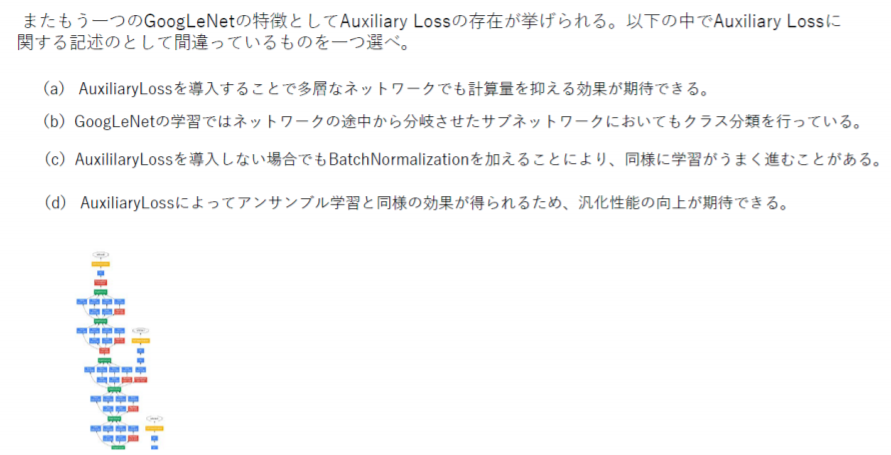
googLenet

1. 1×１畳み込みによる次元削減

下図の例のように、黄色の１×１フィルタの追加によって、次元を削減して２チャンネルの入力画像でも、出力は①と同じサイズになり、パラメータ数の減りによって、計算効率が向上される。



[p.9]

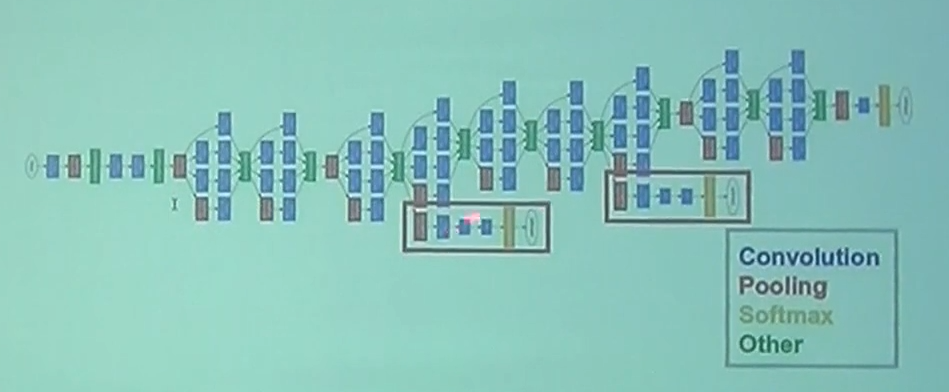


■考察

（a）が間違っている。AuxiliaryLossを導入することで、計算量が増える。

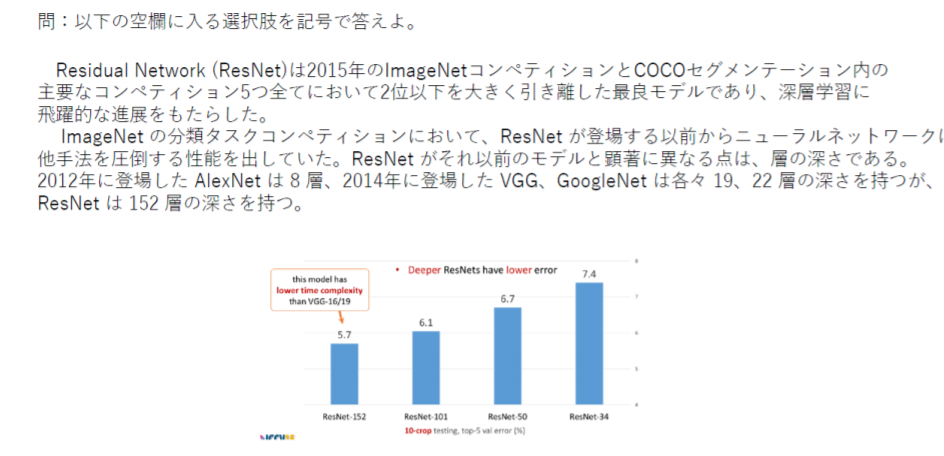
GoogLenetに、Inception以外、二つ赤枠（Auxiliary classfiers）が追加されている。

Auxiliary classfiersは、ＧoogLeNetを学習するときに、同時に学習するので、自分の勾配を直接最後の勾配にフィードバックする。ＲＥＳＮｅｔと外観が違うが、同様に勾配消失問題を解決する。ＧoogLeNetを利用するとき、ＧoogLeNetより外す。

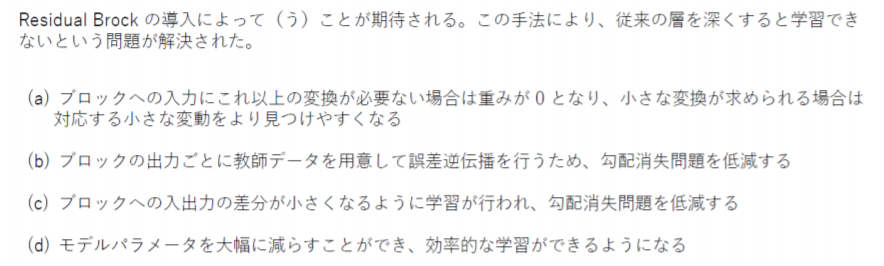


1. 上記の図によって、１×１畳み込みによる次元が削減し、計算量を抑えるが、Ａuxiliary Classfiersの追加が計算量を減少することを期待できない。
2. Auxiliary Classfiersをサブネットワークとして、学習を参加して、クラスを分類して、GoogLeNetに勾配を貢献することを期待する。
3. BatchNormalizationは、勾配消失の対応策のひとつなので、AuxiliaryClassifierを追加しなくても、勾配消失の問題を改善できる。
4. アンサンブル学習は、複数の学習器を用意して、それらの予測結果を統合して半化性能を高めることである。Auxiliary Classfiersは別の学習器として使用し、複数の学習器で、汎化性能を高めることを期待できる。

【ｐ１０】







■考察

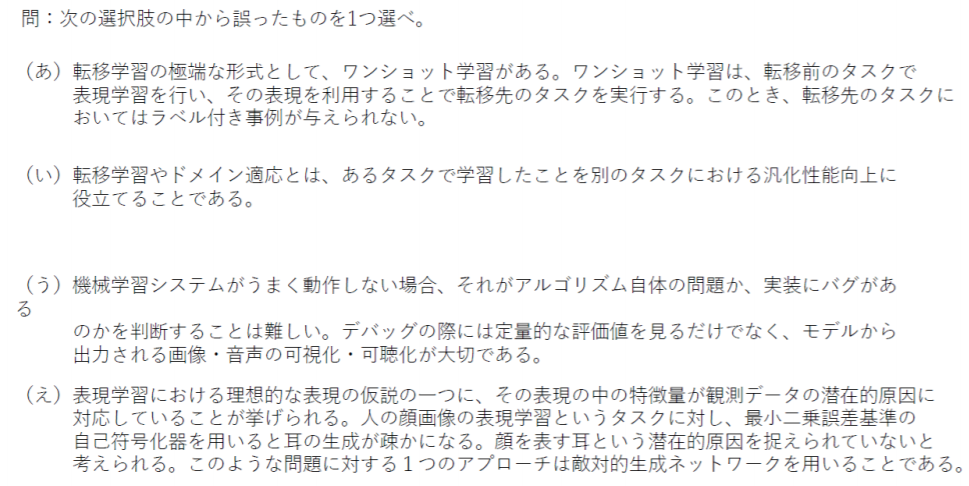
（あ）は(a)勾配消失が正解である。

(い)は（a）Identity mappingが正解である。Idenitity mappingは恒等写像であり、

残差を学習でき、勾配消失が解消できる。

（う）は（a）が正解である。ブロックへの入力が必要ない場合は、重みが０のため、そのままの値を通すことができる。

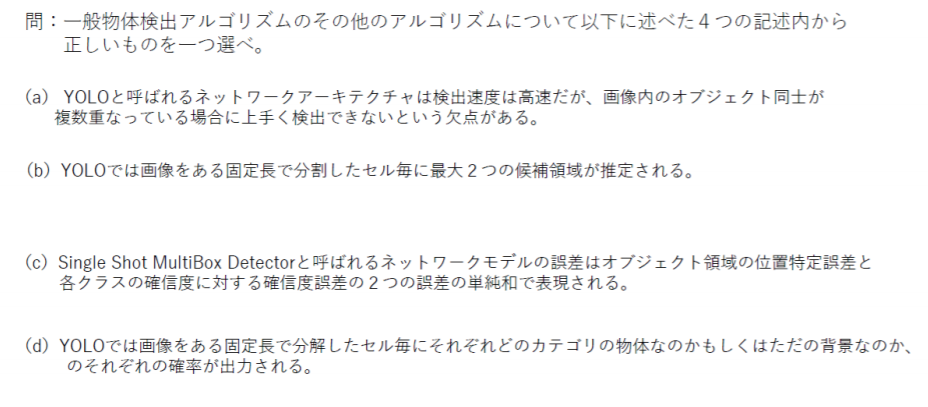
【ｐ１３】



■課題

（あ）が誤ったものである。

【ｐ１４】



■課題

（a）が正しい。

　(b)は最大２つの候補領域というのが誤りで、必ず２つの候補領域がある。

　(c) は２つの誤差の単純和というのが誤りで、位置特定誤差と各クラスの確信度に対する　　２つの誤差の重み付けまで表現される。

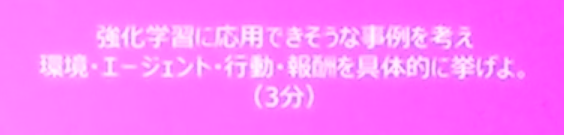


■課題

ＶＧＧの特徴は、2014年のモデルで、Conv-Conv-Poolという単純なネットワークの積み重ねでできているため、ネットワークの構成がシンプルだが、パラメータが多い。

GoogLeNetの特徴は、inception moduleを使っており、1×１の畳み込みを使った次元削減や、様々なフィルタをつかったスパース化である。

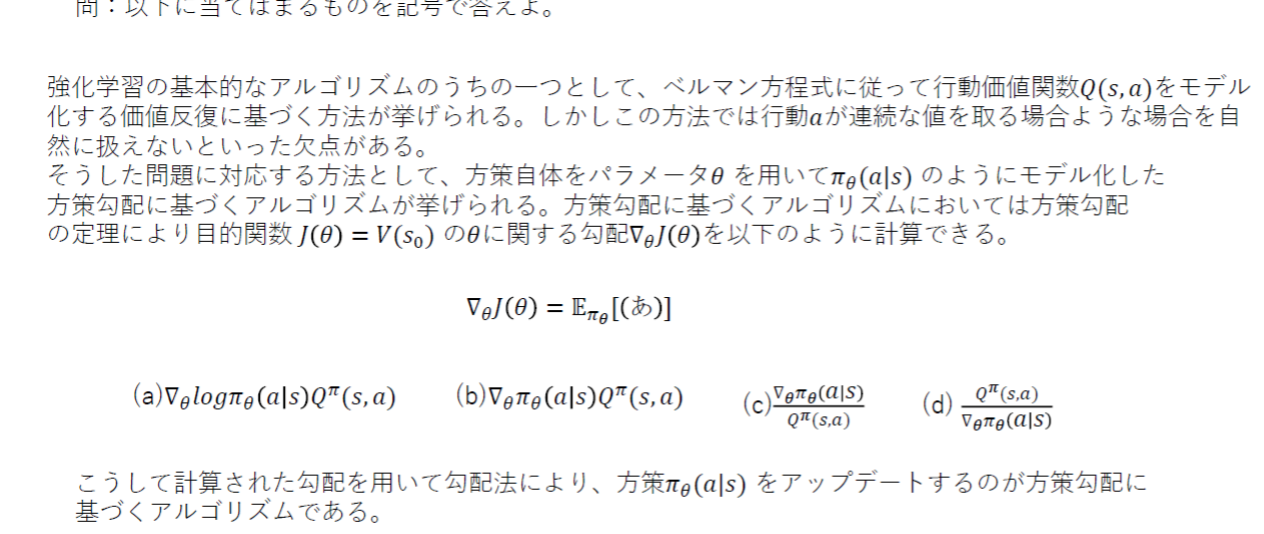
ResNetの特徴は、skip connection identity moduleを使うことで、残差接続を行い、深い層でも学習が行えることである。



■課題

例として、環境はオセロゲームの盤面、エージェントはプレイヤー、行動はどこに石を置くか、報酬は勝率を上げることである。

【ｐ３６】



■課題

（a）が正解である。

**演習考察**

[**DN65\_線形回帰**](https://vimeo.com/showcase/5647400/video/308284599)[**DN66\_非線形回帰**](https://vimeo.com/showcase/5647400/video/308265250)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/4_1_tensorflow_codes-ensyu.pdf>

■考察（線形回帰について）

・noiseの値を 小さくすると（0.3を0.03に変更）、訓練誤差が小さくなり、回帰曲線の予測精度が良くなる。

・ｄの数値を変えても（３を３０に変更）、訓練誤差はほぼ変わらず、回帰曲線を問題なく予測できる。

■考察（非線形回帰について）

・noiseの値を 小さくすると（0.05を1に変更）、訓練誤差が大きくなり、回帰曲線の予測精度が悪くなる。

・ｄの数値を変えると、回帰曲線の形は変わるが、問題なく回帰曲線を予測できる。

■考察（y=30x2+0.5x+0.2の回帰について）

・学習率が0.001のときは学習が進まなかったが、学習が進むように学習率を0.1に変更すると学習が進み、予測精度が良くなった。

・上記では学習率を変えたが、エポック数を10000から50000に変えることで、予測精度を改善できる。

■考察（分類3層について）

・分類1層（中間層無し）から分類３層（中間層２層）にすると、表現力があがり、正解率が上がった。（8６％→９０％）

・エポック数を増やすと、学習が進み、正解率が上がった。（90％→92％）

■考察（Dropout\_ratioについて）

Dropout\_ratioを0.5から０に変更すると、学習の進み具合が早くなり、テスト誤差が早く収束する。Dropout\_ratioが大きいほうが汎化性能が上がり、テスト誤差が小さくなる。

[**DN74\_Keras1**](https://vimeo.com/showcase/5647400/video/308324128)**、**[**DN76\_Keras3**](https://vimeo.com/showcase/5647400/video/308287541)

**■演習実施結果**

<https://github.com/yamda2000/study/blob/master/4_3_keras_codes-ensyu.pdf>

■考察（単純パーセプトロンについて）

・random\_seed=0を変えない場合は乱数が固定されるが、変えると異なる乱数に固定される。

・エポック数を30から１００に変更すると、LOSSが減り、学習が進んでいることがわかる。

■考察（ 分類 (iris)について）

・活性化関数をReLUからSigmoidに変えると、訓練誤差は小さくなるが、テスト誤差は大きくなる。

・活性化関数がReLUで、SGDの学習率を０．１から１に上げると学習速度が速くなるので、精度があがる。

■考察（分類 (mnist)）

・one\_hot\_vector化する場合は、誤差関数はcategorical\_crossentropy、one\_hot\_vector化しない場合は、誤差関数はsparse\_categorical\_crossentropyを使用する。

・Adamの学習率を0.001から0.1に上げると、値が大きすぎるため、収束しなかった。

■考察（2進数足し算の予測について）

・SimpleRNNの隠れ層のユニット数を１６から１２８に変更すると、表現力があがり、

　予測精度が上がる。

・SimpleRNNの隠れ層の活性化関数をReLUからsigmoidに変更すると、勾配消失が起こるため、学習が進まず、予測精度が悪くなる。

・SimpleRNNの隠れ層の活性化関数をReLUからtanhに変更すると、学習の進行具合は変わらず、予測精度はほぼ同じである。

・SimpleRNNの最適化手法をSGDからAdamに変更すると、学習が進み、予測精度があがる。

・Dropoutを有効にすると、汎化性能はあがるが、学習の進みが遅くなる。

・unrollはsimpleRNNをスピードアップできるが、展開は短い系列にのみ適している。