Vol.2015-CVIM-195 No.53
2015/1/23

Stacked Denoising Autoencoder の汎化性能向上に関する一検討

寺本隆磨[†1]　長田茂美[†1]

概要：Stacked Denoising Autoencoder(SdA)は，Denoising Autoencoder(dA)と呼ばれる Autoencoder を多層に積み重ねて構成した Deep Neural Network の一種である．ノイズを付加した画像からノイズを除去した画像を復元できるように Pre-training する dA は，画像からその主成分を頑健に抽出することが可能となり，SdA は高い汎化性能を発揮できる．しかし，一方で，SdA には，dA を Pre-training する際のノイズ量が少なすぎると汎化性能が不足したり，逆に多すぎるとうまく収束しなかったりと，パラメータの調整が難しいという問題点がある．本研究では，dA の Pre-training の際に付加するノイズの量に着目し，学習の進行に応じてそのノイズ量を適応的に制御し，SdA の汎化性能を向上させることを目的として，dA に付加するノイズ量が SdA の認識精度に及ぼす影響についての評価実験を実施した．本稿では，その評価実験および考察を述べる．

1. はじめに

近年，Deep Learning と呼ばれる Deep Neural Network (DNN)を用いた機械学習の方法論や一般物体認識，音声認識などへの応用が盛んに研究されており，数々のベンチマークテストで高い認識精度を示している [2) 4)]．従来からの誤差逆伝播学習などの教師あり学習を用いた多層の DNN の学習では，層数が増えると局所最適解にトラップされたり，学習誤差が十分に下層に伝搬されなかったり，さらには，高い表現能力を有するがゆえに過学習が生じるなど，DNN の汎化性能が著しく低下するという問題があった．

このような問題を解決するために，Hinton らは，多層の DNN を構成する各層ごとに事前に教師なし学習を行なう Pre-training という方法を導入し，多層の DNN の学習に有効に機能することを示した [5)]. Pre-training を実行した層は，入力データからある種の特徴量を抽出する能力を有することが分かっており [6)]，このことは，Pre-training によって入力データに適した特徴抽出器が自動的に構築されることを意味している．したがって，これらの事前に Pre-training を実行した層を多層に積み重ね，最終的に Fine-tuning と呼ばれる教師あり学習を実行することにより構築した DNN は，局所最適解にトラップされにくく，最適解により近い解に到達できるため，より高精度な認識を実現できる．

この Pre-training は, Bengio らが提案している Autoencoder を多層に積み重ねた Stacked Autoencoder(SA)においても有効に機能することが示されている [1)]．ここで用いられた Autoencoder は, 入力データそのものを教師データとして教師あり学習を行なう 3 層の Neural Network である．また，Vincent らによって, 入力データにノイズを付加したデータからノイズを除去した元のパターンを復元することを Pre-training した Denoising Autoencoder[7)]を用いて構成した Stacked Denoising Autoencoder（SdA）が提案されており，単純な Autoencoder を用いた SA よりも高い汎化性能を実現できることが実験的に示されている [8)].

しかし，一方で，SdA には，SdA には，dA を Pre-training する際に付加すべきノイズの量が少なすぎると汎化性能が不足したり，逆に多すぎるとうまく収束しなかったりと，パラメータの調整が難しいという問題点がある．

そこで，本研究では，SdA を構成する dA の Pre-training の際に与えるノイズ量に着目し，学習の進行に応じてそのノイズ量を適応的に制御して，SdA の汎化性能を向上させることを目的として，dA に付加するノイズ量が SdA の認識精度に及ぼす影響についての評価実験を実施した．本稿では，その評価実験の結果と考察を述べる．

2. Stacked Denoising Autoencoder

ここでは，Stacked Denoising Autoencoder（SdA）の構成要素である Autoencoder と Denoising Autoencoder について述べる．

2.1 Autoencoder

Autoencoder は, 入力データそのものを教師データとして教師あり学習を行なう 3 層の Neural Network のことである．入力データ自体を教師データとすることから，Autoencoder の外部から見れば教師なし学習とみなすことができる．図 2.1 にその概要を示す.

†1 金沢工業大学 大学院 工学研究科 情報工学専攻
Kanazawa Institute of Technology

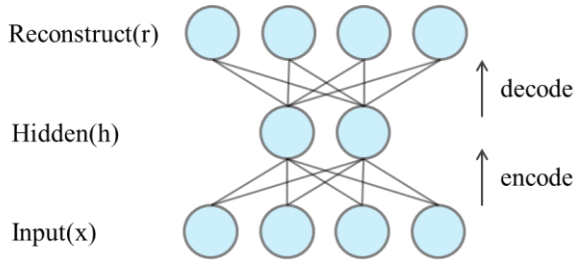

図 2.1　Autoencoder 概要

入力層への入力を x, 隠れ層の出力を h, 入力層と隠れ層との間の重みとバイアス項をそれぞれ W と b とすると, それらの間には次式が成り立つ.

$$h = f(Wx + b) \tag{1}$$

同様に，出力層の出力 r は，隠れ層の出力 h と隠れ層と出力層との間の重みとバイアス項W'とb'を用いて，次式で与えられる.

$$r = f'(W'h + b') \tag{2}$$

この r が x と可能な限り近くなるように W と b，W'と b'を決定することが，Autoencoder の学習となる．W と W'に関して，W'を W^{T} と制約する場合がある．活性化関数 f には

$$f(x) = \tanh(x) \tag{3}$$

や，sigmoid 関数

$$f(x) = \frac{1}{1+\exp(-x)} \tag{4}$$

が使用されるが, 近年では, 微分が容易で, 高速な Rectified Liner Unit（ReLU）

$$f(x) = \max(0, x) \tag{5}$$

がよく使用されている.

Autoencoder は, 隠れ層のユニットの数が多いほど表限能力が高まり，出力層での入力の再現性も同様に高まる．一方で，パターン認識タスクに用いる場合，表現能力の高さが逆に不利に働くことがある．出力層において入力データをほぼ完ぺきに再現できてしまうと，学習データに対する過学習を引き起こしやすくなり, Autoencoder の汎化性能は低下してしまう．この過学習を防ぐために，ユニット数にボトルネックを設ける, スパース正規化 [3]を行なうことで, スパースな特徴を抽出する層が学習され，汎化性能の高い SA を構築することが可能となる．ここで，ユニット数のボトルネックが大きすぎると，表現力の低下から入力の再現が不十分になり，精度に悪影響が出るため，最適なユニット数の選択は，Autoencoder の構築において，重要な要素であるといえる.

2.2 Denoising Autoencoder

Denoising Autoencoder（dA）は，上述した Autoencoder の学習段階で，入力データにノイズを付与し，そのノイズが付与する前のオリジナルデータを復元するように学習を行なう Autoencoder である．このようにして，dA は，単純な Autoencode と比較して, より頑健な特徴抽出が可能となり，未知のデータに対する汎化性能高めることができる．ノイズを付与する方法として，入力データの特定要素の値に 0 を乗算する Binomial がある．Binomial は，ベルヌーイ分布に基づいて入力にノイズを付与する方法である．ノイズを付与しないという事象と，ノイズを付与するという事象の 2 つが存在しており，ノイズを付与しない確率を p, ノイズを付与する確率を$(1-p)$とすると，1 回試行した時の確率関数 $P[X = k]$ は，以下のようになる.

$$P[X = k] = p^k(1-p)^{1-k}, k = 0,1 \tag{6}$$

この確率関数に従って入力の値に 1 か 0 を乗算するか，つまり，ノイズを付与するか，入力の値を維持するかが，入力データの次元数分の回数判定される.

ノイズの付与量は, dA の復元性能に大きな影響を与える. 図 2.2 に, 手書き数字画像データセット MNIST のデータに異なる条件でノイズを付与して学習させた dA からオリジナルデータを復元させた結果を示す.

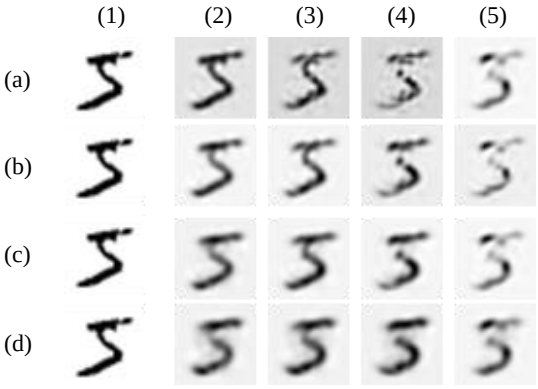

図 2.2　dA による画像復元．(1) オリジナル画像，(2) 復元前のノイズ付与量 0.0，(3) 復元前のノイズ付与量 0.25，(4) 復元前のノイズ付与量 0.5，(5) 復元前のノイズ付与量 0.75，(a) 学習時のノイズ付与量 0.0，(b) 学習時のノイズ付与量 0.25，(c) 学習時のノイズ付与量 0.5，(d) 学習時のノイズ付与量 0.75.

図 2.2 の(a)から(d)の行は，学習条件，つまり，学習時に付与したノイズの量によって分けられており， (2)から(5)の列は，テストデータを復元する前にノイズを付与した量により分けられている. 上から下の行にいくほど, 学習時に, 学習データに付与したノイズ量が多くなり，左から右の列に行くほど，復元時にテスト画像に付与したノイズ量が多くなる．これらの結果から，学習時にノイズの付与量が少ない場合，復元時に加えたノイズの影響を大きく受け，うまく復元できていないことが分かる．また，学習時にノイズ付与量が多い場合，復元時に加えたノイズの影響を大きく受けず，復元できている一方で，ノイズが少ない画像を復元する際の復元性能は低下していることが確認できる.

2.3 Stacked Denoising Autoencoder

Stacked Denoising Autoencoder（SdA）は，図 2.3 に示すように，上述した学習（Pre-training）を行なった dA の層を多層に積み重ねて構築した DNN の一種である．SdA を構築する場合, Pre-training された dA をいくつか積み重ね，最上位層にもう 1 層を追加して Multi-Layer-Perceptron として教師あり学習により Fine-tuning するタイプと，最上位層までのすべての層を dA で構成し，全体を 1 つの大きな dA とみなして教師なし学習により Fine-tuning するタイプの SdA が存在する．前者の Fine-tuning は，Pre-training 済みの層は数値を固定して改めて学習を行なわない場合と，改めてすべての層を学習する場合の 2 つの手段をとることができる．一般に，適用分野に関しては，前者は一般物体認識におけるクラス分類に，後者は画像ノイズ除去や欠損補間に用いられている. いずれの場合も, dA を Pre-training し，その出力を入力とする dA を新たに学習する，という動作を繰り返す必要がある。そのため, 層数が増えればその分，学習にかかる時間も増加することになり，いかに最適なパラメータを決定するかが極めて重要となる．

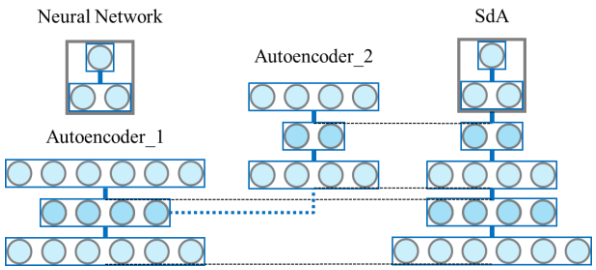

図 2.3 Stacked Denoising Autoencoder

3. 実験

SdA によるパターン認識タスクにおいて，Autoencoder として dA を用いることで，未知のデータに対する汎化性能の向上を図ることができる．しかし，ノイズを多く付与すればコンスタントに汎化性能が向上するというわけではなく，あるノイズ量を境に SdA の認識精度は低下してしまう．また，認識精度の向上に有効な最適なノイズ量を Pre-training の前に推定することは難しく，現状は試行錯誤的に決定せざるを得ない．

そこで，Pre-training の初期段階では，発散を回避するために学習データに付与するノイズ量を低減し，学習が進むにつれてノイズ量を増加させるという戦略が，汎化性能の向上に有効であるとの仮説を立て，以下の 3 種類の実験を行なった．

実験 1：ノイズ量ごとの精度評価　Pre-training の際に付与するノイズ量は各層，学習 epoch で固定として，dA および SdA の学習を行なったのち，SdA の認識精度を評価する．その後，ノイズ量の値を変え，学習と評価を繰り返す．

実験 2：ノイズ量を変化させた場合の精度調査　条件は**実験 1** と同じとし，ノイズ量のみを学習 epoch が進むごとに増加させた場合，減少させた場合両方の精度を評価する．そして，ノイズ量の，変化を開始する数値と変化を終了する数値を変え，それぞれ比較する．

実験 3：ノイズ量の変化方法の比較　**実験 2** では 1epoch ごとに一定量，ノイズ量を増減させていた．そこで，ノイズ量の変化方法を変更し，それぞれの精度を比較することで，適したノイズの制御方法を探る材料とする．

3.1 実験方法

実験に用いる画像データセットとして，MNIST と CIFAR-10 を選定した．MNIST は 0 から 9 までの手書き数字を，28×28pixel，1pixel の値が 0-255 の画像としてまとめたものである．配布の段階で学習用データセットと，テスト用データセットの 2 つに分割されており，学習用データセットに 6,0000 枚，テスト用データセットに 1,0000 枚の画像が含まれている． CIFAR-10 は，10 カテゴリの物体の写真を，32×32 のカラー画像としてまとめたもので，MNIST と同じく，学習用データセットと，テスト用データセットに分割されて配布されている．学習用データセットに 50,000 枚，テスト用データセットに 10,000 枚の画像が含まれている．今回の実験では dA，SdA の学習に学習用データセットを，SdA の精度テストにテスト用データセットを用いた. MNIST のデータは正規化以外の加工をせず実験に使用し，CIFAR-10 のデータは，データセット内でコントラストの正規化を行なったのち，実験に用いた．

dA の各種パラメータに関しては, 基本的には実装に用いた pylearn2 のデフォルトに基本的に準拠している．ノイズの付与には Binomial を, 活性化関数には tanh を用いた．**2.1** で述べたとおり，dA の学習時に，W'を W^{T} と制約する場合があるが，今回はデフォルトのパラメータを採用し，その制約は設けていない．

実験の際に構築する SdA は，dA を積み重ね，その最上層に入力層のユニット数が，前層の隠れ層のユニット数と等しい, 出力層に 10 ユニットを有する Softmax 層を結合する構成とした．また，Fine-tuning の際は，デフォルト設定である全層のユニットを更新する手法を採用した．

実験 1 の評価実験は，MNIST と CIFAR-10 を対象に行なった. この際, dA, SdA の学習は 400epoch 行なった. MNIST の実験に用いる dA は，入力層のユニット数が，MNIST の画素数と等しい 784 ユニット，隠れ層のユニット数が 196 ユニットのものと，入力層に 196 ユニット，隠れ層に 64 ユニットをもつものを用意し，Pre-training を行なった．CIFAR-10 の実験には，4 層の dA を用意した．1 層目は，入力層に 3072 ユニット，隠れ層に 1728 ユニットを有し，2

層目は入力層に 1728 ユニット, 隠れ層に 768 ユニットを有する．3 層目は入力層に 768 ユニット，隠れ層に 192 ユニットを有し，4 層目は入力層に 192 ユニット，隠れ層に 48 ユニットを有する．**実験 2** に関しては，データセットを MNIST に絞り，dA の学習 epoch を 400 に加え，40，10 の 3 種とし，1epoch 学習が進行するごとに定数をノイズ量に加え，精度の変化を調査した．この定数は，学習開始時のノイズ量と，変化終了時のノイズを決定したのち，その値域を学習 epoch 数で割った値としている．つまり，epoch 数が 40，学習開始時のノイズ量が 0.0，学習終了時のノイズ量が 0.2 だとすると，epoch ごとに 0.005 ずつノイズ量が増加するということである． **実験 3** は，**実験 2** の変化方法と異なる，7 種の変化方法を**実験 2** の 40epoch 時において最高精度を得られた条件に適用して精度の変化を調査する．

3.2 実験結果および考察

実験 1 の結果を，表 3.1 および表 3.2 に示す．この結果からは，CIFAR-10 に関してはノイズ量が 0.2，MNIST の場合には 0.55 のところに精度のピークが存在することが確認できる．最も注意すべきは，そのピークの値が 2 つのデータセット間で大きく離れていることである．ノイズ量の自動調節機能には，データセットごとにパラメータを手動で調整するのではなく，システムのみでこの差を埋める機能を組み込む必要があることが判明した．

次に，**実験 2** の評価を行なうが，その前に，**実験 2** に際して，事前に行なった予備実験の結果を表 3.3 に示す．

表 3.3　ノイズ量固定時の最高精度

	10	40		400
ノイズ量	0.7	0.58	0.59	0.52
精度	0.9454	0.9479	0.9479	0.9523

これは，**実験 2** の評価に用いるため，学習 epoch ごとに，ノイズ量固定時の最高精度を，厳密に求めたものである．epoch 数が 10 の場合は 0.7 が, epoch 数が 40, の場合は 0.58 と 0.59 が，epoch 数 400 の場合は 0.52 がノイズ量を固定とした際の最高精度を得られるノイズ量である．

実験 2 の精度を表 3.4 から表 3.6 に示す. 表 3.4 は epoch 数が 400, 表 3.5 が 40, 表 3.6 が 10 の結果である．400epoch の実験に関しては，ノイズ量を減少させた際に，固定時の最高精度を上回っていることが確認できるが，40，10epoch の場合には，逆にノイズ量を減少させた際に精度が向上するケースは無く，ノイズ量を増加させた場合に，固定時の精度を上回っている場合がいくつか確認できる．

次に**実験 3** の評価を行なう．ノイズ量の変化範囲は，**実験 2** より，0.35 から 0.75 とした．次に，図 3.1 に，ノイズ量をどのように変化させたかを示し，表 3.7 にその精度を示す．

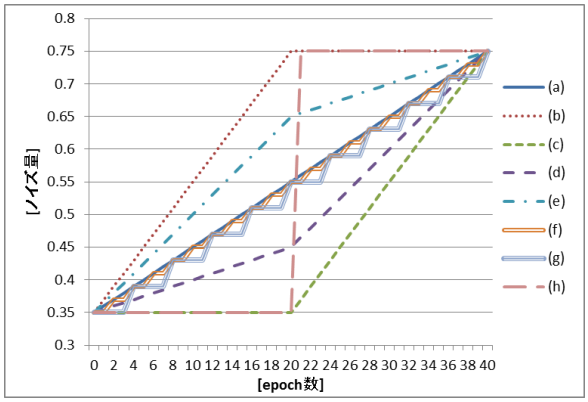

図 3.1　ノイズ量の変化方法

表 3.7　ノイズ量変化方法変更時の精度

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
精度	0.9485	0.9463	0.9457	0.9481	0.9488	0.9475	0.9478	0.9466

表 3.7 の(a)は，**実験 2** の変化方法であり，指標として記載している．(f)，(g)の結果から，**実験 2** のように，1epoch 間隔でノイズ量を変化させる方法が，精度が良くなるということが分かる．また，学習が進むにつれ，ノイズの変化量を減少させていく制御方法が，高精度につながるということもこの実験結果から判断できる．一方で，学習開始時に大きくノイズを増加させてしまうと，(b)のように，逆に精度が低下してしまうことが分かる．むしろ，20epoch までの変化量が小さい(d)の方が(b)と比較して高精度であるため，ノイズ量を制御する場合は，ノイズを加える量の変化は小さい方がよいことが判断できる．

表 3.1　CIFAR-10　ノイズ量固定時の精度(epoch:400)

ノイズ量	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
精度	0.3873	0.3894	0.3888	0.3847	0.3895	0.3848	0.3843	0.3843	0.375	0.3729	0.3738

表 3.2　MNIST　ノイズ量固定時の精度(epoch:400)

ノイズ量	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7
精度	0.9323	0.9329	0.9335	0.9373	0.9403	0.9428	0.9438	0.9466	0.9474	0.95	0.9501	0.9519	0.9495	0.948	0.946

表 3.4　MNIST　ノイズ量変化時の精度(epoch:400)

[開始時のノイズ量] ＼ [最終的なノイズ量]	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7
0	0.9323	0.9321	0.9331	0.9330	0.9341	0.9370	0.9409	0.9429	0.9445	0.9456	0.9446	0.9473	0.9465	0.9472	0.9471
0.05	0.9315	0.9329	0.9335	0.9342	0.9363	0.9395	0.9423	0.9416	0.9443	0.9464	0.9482	0.9506	0.9472	0.9481	0.9474
0.1	0.9327	0.9320	0.9335	0.9350	0.9382	0.9422	0.9426	0.9448	0.9455	0.9470	0.9476	0.9491	0.9486	0.9478	0.9477
0.15	0.9323	0.9329	0.9337	0.9373	0.9407	0.9418	0.9423	0.9450	0.9475	0.9501	0.9477	0.9491	0.9497	0.9497	0.9472
0.2	0.9327	0.9338	0.9365	0.9392	0.9403	0.9425	0.9451	0.9452	0.9478	0.9477	0.9486	0.9502	0.9505	0.9499	0.9487
0.25	0.9324	0.9351	0.9394	0.9410	0.9417	0.9428	0.9445	0.9467	0.9486	0.9497	0.9492	0.9503	0.9491	0.9498	0.9507
0.3	0.9352	0.9370	0.9397	0.9419	0.9422	0.9434	0.9438	0.9457	0.9492	0.9505	0.9501	0.9502	0.9501	0.9486	0.9478
0.35	0.9351	0.9378	0.9400	0.9409	0.9423	0.9451	0.9460	0.9466	0.9478	0.9483	0.9488	0.9492	0.9498	0.9490	0.9499
0.4	0.9355	0.9393	0.9410	0.9423	0.9434	0.9441	0.9468	0.9480	0.9474	0.9498	0.9507	0.9498	0.9503	0.9484	0.9470
0.45	0.9379	0.9413	0.9425	0.9432	0.9448	0.9453	0.9462	0.9489	0.9497	0.9500	0.9517	0.9506	0.9478	0.9481	0.9477
0.5	0.9384	0.9411	0.9432	0.9443	0.9464	0.9480	0.9494	0.9502	0.9486	0.9503	0.9501	0.9489	0.9499	0.9493	0.9495
0.55	0.9386	0.9419	0.9436	0.9469	0.9458	0.9469	0.9506	0.9492	0.9516	0.9529	0.9524	0.9519	0.9507	0.9503	0.9451
0.6	0.9400	0.9421	0.9438	0.9446	0.9463	0.9488	0.9488	0.9491	0.9509	0.9508	0.9514	0.9503	0.9495	0.9458	0.9450
0.65	0.9399	0.9418	0.9438	0.9458	0.9468	0.9482	0.9494	0.9504	0.9510	0.9510	0.9515	0.9514	0.9495	0.9480	0.9469
0.7	0.9412	0.9410	0.9445	0.9455	0.9472	0.9490	0.9509	0.9509	0.9509	0.9514	0.9514	0.9499	0.9495	0.9491	0.9460

→固定時の精度を上回ったケース

表 3.5　MNIST ノイズ量変化時の精度(epoch:40)

[開始時のノイズ量] ＼ [最終的なノイズ量]	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8
0	0.9360	0.9365	0.9369	0.9366	0.9370	0.9381	0.9393	0.9405	0.9415	0.9427	0.9438	0.9443	0.9443	0.9447	0.9448	0.9458	0.9458
0.05	0.9357	0.9368	0.9369	0.9368	0.9372	0.9387	0.9393	0.9411	0.9421	0.9427	0.9444	0.9447	0.9443	0.9443	0.9456	0.9463	0.9462
0.1	0.9362	0.9367	0.9363	0.9374	0.9380	0.9386	0.9396	0.9411	0.9422	0.9425	0.9446	0.9447	0.9452	0.9461	0.9465	0.9461	0.9457
0.15	0.9359	0.9366	0.9367	0.9376	0.9384	0.9389	0.9399	0.9411	0.9422	0.9435	0.9442	0.9453	0.9450	0.9467	0.9463	0.9460	0.9465
0.2	0.9363	0.9364	0.9372	0.9375	0.9384	0.9393	0.9405	0.9411	0.9418	0.9448	0.9455	0.9462	0.9465	0.9460	0.9465	0.9468	0.9464
0.25	0.9361	0.9369	0.9373	0.9381	0.9387	0.9398	0.9410	0.9411	0.9429	0.9448	0.9461	0.9461	0.9465	0.9474	0.9471	0.9476	0.9473
0.3	0.9362	0.9367	0.9372	0.9378	0.9387	0.9402	0.9409	0.9416	0.9437	0.9453	0.9453	0.9467	0.9465	0.9480	0.9472	0.9481	0.9478
0.35	0.9361	0.9368	0.9377	0.9377	0.9390	0.9406	0.9408	0.9424	0.9440	0.9456	0.9465	0.9463	0.9476	0.9485	0.9475	0.9485	0.9483
0.4	0.9365	0.9367	0.9370	0.9380	0.9394	0.9401	0.9417	0.9430	0.9445	0.9464	0.9473	0.9471	0.9469	0.9476	0.9484	0.9472	0.9472
0.45	0.9361	0.9369	0.9373	0.9383	0.9402	0.9406	0.9420	0.9439	0.9451	0.9474	0.9479	0.9471	0.9474	0.9481	0.9480	0.9468	0.9466
0.5	0.9362	0.9370	0.9375	0.9388	0.9403	0.9412	0.9430	0.9436	0.9456	0.9466	0.9464	0.9477	0.9479	0.9483	0.9464	0.9445	0.9457
0.55	0.9363	0.9367	0.9379	0.9387	0.9406	0.9419	0.9436	0.9433	0.9461	0.9477	0.9466	0.9476	0.9477	0.9473	0.9462	0.9446	0.9453
0.6	0.9359	0.9368	0.9380	0.9389	0.9408	0.9430	0.9429	0.9447	0.9467	0.9469	0.9467	0.9470	0.9477	0.9473	0.9462	0.9464	0.9447
0.65	0.9358	0.9370	0.9377	0.9399	0.9415	0.9431	0.9436	0.9444	0.9464	0.9456	0.9466	0.9478	0.9465	0.9461	0.9454	0.9437	0.9440
0.7	0.9363	0.9376	0.9384	0.9401	0.9420	0.9429	0.9434	0.9448	0.9458	0.9456	0.9465	0.9479	0.9462	0.9460	0.9461	0.9445	0.9437
0.75	0.9373	0.9377	0.9386	0.9404	0.9420	0.9426	0.9429	0.9458	0.9458	0.9467	0.9476	0.9475	0.9465	0.9466	0.9449	0.9447	0.9428
0.8	0.9368	0.9374	0.9390	0.9402	0.9422	0.9429	0.9438	0.9452	0.9464	0.9471	0.9469	0.9474	0.9474	0.9462	0.9452	0.9447	0.9452

→固定時の精度を上回ったケース

表 3.6　MNIST ノイズ量変化時の精度(epoch:10)

［開始時のノイズ量］	［最終的なノイズ量］																	
	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8
0	0.9376	0.9381	0.9386	0.9390	0.9393	0.9393	0.9394	0.9402	0.9401	0.9410	0.9417	0.9420	0.9425	0.9432	0.9438	0.9446	0.9456
0.05	0.9377	0.9382	0.9387	0.9388	0.9392	0.9391	0.9397	0.9394	0.9403	0.9411	0.9415	0.9418	0.9428	0.9432	0.9439	0.9447	0.9453
0.1	0.9379	0.9383	0.9385	0.9386	0.9389	0.9393	0.9398	0.9399	0.9403	0.9405	0.9417	0.9418	0.9425	0.9431	0.9441	0.9449	0.9450
0.15	0.9378	0.9382	0.9384	0.9388	0.9389	0.9396	0.9394	0.9395	0.9400	0.9405	0.9417	0.9419	0.9423	0.9428	0.9436	0.9449	0.9455
0.2	0.9377	0.9381	0.9385	0.9389	0.9391	0.9394	0.9389	0.9397	0.9401	0.9407	0.9415	0.9416	0.9419	0.9426	0.9432	0.9445	0.9457
0.25	0.9378	0.9381	0.9386	0.9388	0.9390	0.9392	0.9391	0.9399	0.9398	0.9406	0.9419	0.9419	0.9422	0.9426	0.9434	0.9448	0.9461
0.3	0.9380	0.9381	0.9385	0.9387	0.9389	0.9390	0.9391	0.9400	0.9400	0.9410	0.9419	0.9419	0.9422	0.9425	0.9430	0.9451	0.9459
0.35	0.9379	0.9382	0.9384	0.9386	0.9386	0.9393	0.9392	0.9397	0.9402	0.9412	0.9419	0.9419	0.9414	0.9425	0.9441	0.9442	0.9460
0.4	0.9379	0.9382	0.9385	0.9387	0.9387	0.9390	0.9396	0.9398	0.9403	0.9413	0.9415	0.9418	0.9416	0.9430	0.9443	0.9447	0.9464
0.45	0.9380	0.9380	0.9383	0.9382	0.9384	0.9391	0.9397	0.9397	0.9405	0.9415	0.9414	0.9424	0.9421	0.9434	0.9443	0.9446	0.9463
0.5	0.9378	0.9383	0.9385	0.9381	0.9387	0.9389	0.9394	0.9397	0.9407	0.9414	0.9415	0.9423	0.9428	0.9443	0.9447	0.9447	0.9457
0.55	0.9375	0.9384	0.9376	0.9384	0.9389	0.9388	0.9389	0.9399	0.9404	0.9412	0.9419	0.9430	0.9435	0.9441	0.9444	0.9455	0.9456
0.6	0.9376	0.9379	0.9380	0.9384	0.9385	0.9383	0.9389	0.9398	0.9405	0.9418	0.9426	0.9433	0.9438	0.9446	0.9446	0.9458	0.9447
0.65	0.9374	0.9377	0.9377	0.9381	0.9382	0.9382	0.9389	0.9399	0.9409	0.9421	0.9423	0.9427	0.9438	0.9446	0.9448	0.9452	0.9448
0.7	0.9375	0.9375	0.9377	0.9376	0.9376	0.9384	0.9397	0.9399	0.9410	0.9420	0.9423	0.9431	0.9439	0.9444	0.9454	0.9450	0.9433
0.75	0.9376	0.9373	0.9375	0.9377	0.9378	0.9385	0.9394	0.9405	0.9409	0.9418	0.9426	0.9432	0.9436	0.9446	0.9442	0.9441	0.9429
0.8	0.9369	0.9375	0.9374	0.9376	0.9377	0.9385	0.9393	0.9398	0.9411	0.9426	0.9431	0.9435	0.9438	0.9433	0.9440	0.9431	0.9426

→固定時の精度を上回ったケース

4. おわりに

本稿では，Stacked Denoising Autoencoder (SdA)の汎化性能の向上を目指して，dA の Pre-training の際に付与するノイズ量が SdA の認識精度に及ぼす影響について評価実験を行なった．その結果，ノイズの付与量が固定の場合の最適値と比較し，dA の学習回数が 40epoch 以下の少ない学習条件下では，ノイズの付与量を徐々に増加することで，微小ではあるが SdA の認識精度の向上が確認できた．しかし，dA の学習回数が 400epoch の場合には，逆に，ノイズの付与量を徐々に増加することが SdA の認識精度の低下を招いている．今回の評価実験では， SdA の Fine-tuning において，dA のパラメータも含めて微調整を行なったが，今後は，dA の Pre-training 時のノイズ付与量が dA 単体の汎化性能に及ぼす影響についても評価し，どの要因が汎化性能向上のキーとなるかを精査するとともに，学習の進行に応じてそのノイズ付与量を適応的に制御し，SdA の汎化性能を向上させる手法を検討していく予定である．

参考文献

1) Bengio, Y., Lamblin, P., Popovici, D. Larochelle, D., Greedy Layer-Wise Training of Deep Networks, NIPS (2007).
2) Cires¸an, D., Meier, U. and Schmidhuber, J., Multi-column Deep Neural Networks for Image Classification, Arxiv preprint arXiv:1202.2745, (2012).
3) Coates, A., Lee, H., Ng1, A-Y., An Analysis of Single-Layer Networks in Unsupervised Feature Learning, AISTATS (2011).
4) Hinton, G., Deng, L., Yu, D., Dahl, G., Abdel-rahman, M., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., Deep Neural Networks for Acoustic Modeling in Speech Recognition, Signal Processing Magazine, (2012).
5) Hinton, G., Osindero, S., Teh, Y.-W., A fast learning algorithm for deep belief nets, Neural Computation, Vol. 18, pp. 1527-1544 (2006).
6) Hinton, G., Salakhutdinov, R., Reducing the Dimensionality of Data with Neural Networks Science, Vol. 313, No. 5784, pp. 504-507, (2006).
7) Vincent, P., Larochelle, H., Bengio, Y., Pierre-Antoine Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, Proc. 25th Int. Conf. Mach. Learn. (ICML¥'08), pp.1096 -1103 (2008).
8) Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol. P.-A., Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, Machine Learning Research, vol. 11, pp. 3371-3408, (2010).