Vol.2020-AVM-108 No.4
2020/2/27

DNN への電子透かし埋め込みの特性調査

小林栄介[1]　酒澤茂之[1]

概要：学習済み深層学習(DNN:Deep Neural Network)モデルの著作権保護を目的とした電子透かし技術が注目されている。本研究では、モデルパラメータが観測できる前提での既存方式において、埋め込みビット数を 256bit から 512bit に増やすために、電子透かし埋め込みレイヤ数を二倍に増加させた。本来の認識タスクへの影響と、電子透かしへの攻撃耐性について調査し、従来の方式との比較を行った結果について報告する。

キーワード：深層学習, 著作権保護, 電子透かし

Performance Analysis of DNN Watermarking

EISUKE KOBAYASHI[†1]　SHIGEYUKI SAKAZAWA[†1]

Abstract: Watermarking to a deep neural network (DNN) is investigated as a copyright protection technique. We extended the conventional method for DNN watermarking to embed double as many bits by using two intermediate layers inside the DNN model. We evaluated two aspects: 1) performance degradation of the original image classification task, 2) attack resilience of the embedded watermarks. Experimental results show that double bits can be embedded without severe drawbacks.

Keywords: Deep Learning, Copyright protection, Watermark

1. はじめに

ディープラーニングは様々な場所での活用が期待されている。例えば、ADAS（先進運転支援システム）では一時停止などの標識や信号機を自動的に認識することができ、さらに歩行者検知をすることで、事故の減少にも役立てられる。医療では、高次元のデータセットによる高精度な顕微鏡システムで、自動的ながん細胞の発見が期待される。

その中心的構成要素である学習済みディープラーニングモデルを作るには、大規模なデータや高性能なパソコン、大量の時間が必要で、学習には様々なコストがかかる。その一方で、再利用が容易である点が挙げられる。そのまま使用することもでき、他にも転移学習などを用いることで個別の目的にチューニングすることができる。よって、権利保護が重要である。

学習済みディープラーニングモデルと知的財産権の関係については、知的財産戦略本部において議論が進められており[1]、学習済みモデルは特許法の要件を満たせば、「方法の発明」、不正競争防止法上の秘密管理性・有用性及び非公知性の要件を満たせば営業秘密として保護され、著作権法の要件を満たせば「プログラムの著作物」として保護される可能性もあるとされている。この権利保護の、技術的な面での対策として、ディープラーニングモデル向け電子透かしの研究が進められている[2]。

本論文の構成は、2 章で従来の電子透かし技術の紹介、3 章ではその埋め込み情報量の拡張と評価方法、4 章では実験結果とまとめの順で説明する。

1 大阪工業大学
Osaka Institute of Technology
2 KDDI 総合研究所
KDDI Research, Inc.

2. ベースとする DNN 電子透かし方式

2.1 電子透かしへの要求条件

電子透かしには、動画像コンテンツ等に対して、条件があり、これはディープラーニングに対しても存在する。以下の表 1 に動画像コンテンツとディープラーニングに対しての比較を示す。

表 1 動画像コンテンツとディープラーニング向け電子透かしの要求条件の比較

	動画像コンテンツ	ディープラーニング
透明性	人の目に見えないこと	本来のタスクに影響がないこと
処理耐性	圧縮などの処理に耐えること	学習済みモデルの再学習等の改変に耐えること
誤検出の抑制	読みだした透かしに誤りがないこと	読みだした透かしに誤りがないこと
埋め込みビット数の確保	多いほど良い	多いほど良い

2.2 電子透かしの埋め込み方式

本研究では、WRN(Wide-Residual-Network)というディープラーニングモデルを使用する。これは、既存研究である DNN(Deep Neural Network)の電子透かしを用いた学習済みモデル保護[2]と同じ条件で行うためである。

今回はディープラーニングモデルに対して、学習している段階でモデルパラータに対して電子透かしを埋め込む形

をとっている。以下に、電子透かしの埋め込み方法について図 1 を示す。

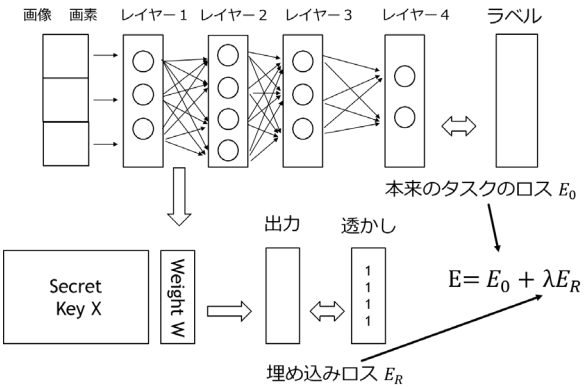

図 1 電子透かしの埋め込み方法

本来の画像認識タスクだけの学習では 1epoch 目の終わりに出力したタスクのロスからバックプロパケーションを通して係数を変化させていく。その工程を何十 epoch も回すことでタスクのロスを減らしている。電子透かしありの埋め込みは電子透かしを埋め込むことできるレイヤの重み係数を取り出し、事前に用意した電子透かしの行列 Secret Key X を掛け合わせる。その出力と、埋め込みたい透かしビットの値の差分から埋め込みロスを算出する。この埋め込みロスの λ(0.01)倍した値と本来のタスクとロスを足すことで、今回のロスとしている。

学習では Cifar-10 をデータセットとして使用する。Cifar10[3]とは airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck の 10 個のクラスからなるデータセットである。画像サイズは 32x32。訓練画像は 50,000 枚(各クラス 5,000 枚)とテスト画像は 10,000 枚(各クラス 1,000 枚)を使用する。合計 6,0000 枚の画像が入っている。

今回の WRN(Wide-Residual-Network)は初期状態から前述のロスを最小化するように学習プロセスを回している。

2.3 電子透かしの検出方法

電子透かしの検出の計算方法はレイヤから取り出した重み係数と電子透かしの行列 Secret Key X を掛け合わせる。そして、その値が 0.5 を超えているとビット 1、0.5 未満ならビット 0 と判定する。

3. 埋め込み情報量の拡張と評価

3.1 埋め込み対象のレイヤ追加

既存研究では、電子透かしの埋め込む情報量が 256bit だったが、今回は 2 個入れることで 512bit に拡張をした。電子透かしの埋め込みロスの計算はレイヤ 2 つ分を純粋に加算した物である。実験に使用するパラメータを図 2 として示す。

データセット	Cifar-10 (60,000 32 × 32 color images, 10classes) 50,000 images for training 10,000 images for test
ネットワークアーキテクチャおよびパラメータ	Wide-Residual-Network[4] (N = 1, k = 4) SGD with Nesterov momentum cross-entropy loss the initial learning rate = 0.1 weight decay = 5.0 x10-4 momentum = 0.9 minibatch size = 64 λ = 0.01
2個の電子透かし	512bit(256bit + 256bit)
2個の埋め込み対象	conv2 group + conv3 group
1個の電子透かし	256bit
1個の埋め込み対象	conv2 group

図 2 今回の実験で使用するパラメータ

3.2 想定する攻撃

今回、攻撃耐性も検証するため、Fine-tuning とモデル圧縮を行う。この両者にはモデルパラメータに影響が出る恐れがあり、モデルパラメータが変化すると、電子透かし自体が消えてしまう可能性があるからである。

まず、Fine-tuning とは既存のモデルの一部を再利用して新しいモデルを構築する手法である。優秀な既存モデルを使い、私用のモデル構築することができ、少ないデータ数で精度の高いモデルを構築できることが利点である。しかし、Fine-tuning はモデルパラメータを変化させてしまう効果があり、今回の研究には悪影響がある。今回は全結合層以外を凍結させており、全結合層のみ再学習させている。データセットは Cifar-10 、モデルは WRN(Wide-Residual-Network)、Epoch 数は 5 で、learning late は 0.01 とする。以下に、Fine-tuning の再学習方法を図 3 として示す。

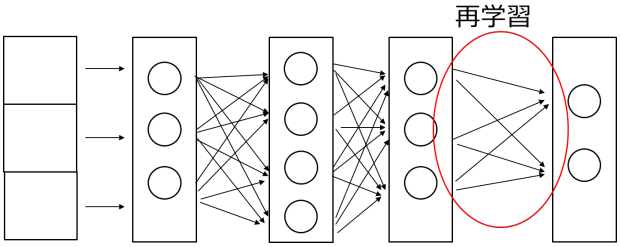

図 3 Fine-tuning で再学習方法

次にモデル圧縮の説明をする。モデル圧縮は精度を保ちながらモデルを軽量化し、処理速度を上げることができる。まず、モデル圧縮には 3 つの方法がある。1 つ目は Pruning といって関係の薄い箇所を 0 にするというものである。これは 0.1~0.3 などの少ししか影響を与えないパラメータ(エッジの重み係数)を 0 にして、そのあと 0 が出来た部分を処理に入れないことで、その影響をあまり与えないパラメー

タの処理を減らすことができる。そうすることで、精度を保ちながらモデルを軽量化することができる。2 つ目は Quantize といってモデルの重みは通常任意の実数値の範囲で計算されるが、それを-1,1 の 2 値に限定する(量子化する)ことで、計算負荷を下げることができる。3 つ目は Distillation といってニューラルネットワークに親と子 2 つを使い、親の結果を子に伝え、子の精度を良くするものである。今回は、1 つ目の Pruning を使う。この Pruning も総合パラメータ数が変化するため、モデルパラメータに変化が起こる。今回はプルーニングの強度の制御方法として Pruning rate というものを設定する。Pruning rate は何%のエッジを pruning するかを決めることができ、今回は 0%を始めとし 5%ずつ最大領域を増やすことで 95%まで設定した時のモデルパラメータの値を調べることができる。データセットは Cifar-10、モデルは WRN(Wide-Residual-Network)、epoch 数は 4、learning late を 0.01 とする。以下に、モデル圧縮の Pruning を図 4 として示す。

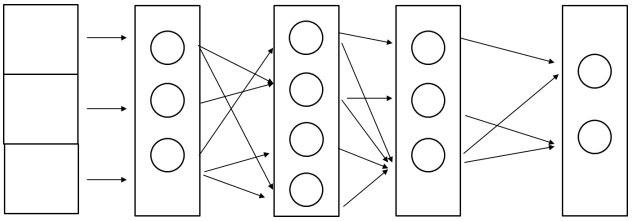

図 4 モデル圧縮の Pruning

3.3 評価方法

まず、電子透かしを 2 個、1 個、なしで学習を行い、モデルパラメータに埋め込んでいく。そして、ディープラーニングモデルを作成し、学習し終わったら本来の画像認識タスクの精度に影響が出るかどうかの比較を出力する。

今回使う画像データセットは Cifar-10 で、これは画像認識で使われているものである。そこに電子透かしを埋め込むのだが、そもそも電子透かしは画像認識で使われている Cifar-10 に対して、影響を与えないように情報を埋め込むものである。よって、電子透かしが 2 個入った場合、本来のタスクの精度にどれだけの影響があるのかを調べる。

次に、攻撃耐性があるかどうかの実験を行い、電子透かしにどの程度影響を与えるかの検証を行う。不正利用者にとって、電子透かしを消してかつ、本来のタスクの精度を維持したままモデルを利用したいと考えるだろう。今回は埋め込む電子透かしの量を増やすことで、逆に攻撃耐性が弱まり、電子透かしの意味がなくなる可能性もあるので、攻撃耐性があるかどうかの実験を行う。

電子透かしの攻撃耐性を調べるために、まず透かしビットすべてを 1 として透かしの埋め込みを行い、その後攻撃を掛けたうえで、式(1)の calculate loss を評価する。values は電子透かしが入っているモデルパラメータが入る。calculate loss は電子透かしのタスクロスである。これはすべての電子透かしが 1 に近づいていると値は小さくなり、逆に 1 から離れていくとこの値は大きくなり、電子透かしに影響を与えていることがわかる。

$$\frac{sum(-\log_e values)}{\text{values.size}} \tag{1}$$

4. 実験結果と考察

4.1 無攻撃時の評価

次の図 5 は電子透かしを 2 個入れた場合と、1 個入れた場合と、なしの場合の本来の画像認識タスクの精度と損失値である。実線グラフが training data に対する損失、破線グラフが validation data に対する損失。最終的なテストデータの具体的な数値が表 2 の数字である。

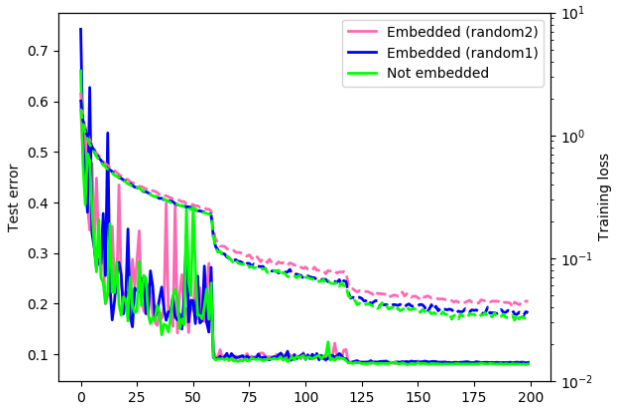

図 5 電子透かしが 2 個、1 個、なしの本来の画像認識タスクの精度の精度(実線 Training error,破線 Test error)

表 2 電子透かしなし、1 個、2 個のテストデータの精度

なし	1個	2個
accuracy= 0.9193	accuracy= 0.9161	accuracy= 0.9168
loss= 0.378759239864	loss= 0.3912966892	loss= 0.374268390775

本来の画像認識タスクの精度について影響がないかを調査した。図 5 より、電子透かしが 2 個の場合と、1 個の場合と、なしの場合の 3 種類を比較のために実験に使用した。その結果、Training loss は 3 種類ともに変化はないことが分かった。しかし、Test error では 2 個入れた場合だけ他の 2 種類との差が少しできたため、電子透かしの数でロス値が増えたことがわかる。表 2 での Accuracy では 3 種類とも精度が高く、loss にもあまり差がなかった。

4.2 ファインチューニング攻撃時の評価

Fine-tuning に対して本来の精度に影響がないかを調べた。表 3 は Fine-tuning を行った時の accuracy と loss である。

表 3 Fine-tuning を行った時の accuracy と loss

accuracy =0.9104
loss = 0.406025373518

表 4 は Fine-tuning をする前とした後の比較を図にしたものである。Fine-tuning は 5epoch で行った。電子透かしの攻撃耐性を調べるために calculate loss を使う。

表 4 Fine-tuning 前後の calculate loss

	レイヤー1	レイヤー2
Fine-tuning前	0.000464444	0.00032883
Fine-tuning後	0.000485801	0.00035098
Fine-tuningとの差（後/前）	1.045984015296	1.067360034060

学習済みモデルに対して Fine-tuning を行い、電子透かしに影響が出ないかの実験を行った。今回は Fine-tuning は全結合層以外を凍結することで再学習をさせた。実験には calculate loss を使い、初期値からどれだけ変化しているかの差を比較した。図 3.3 を比較すると、レイヤ 1 もレイヤ 2 も Fine-tuning されたとしても誤差が小さいので電子透かしは Fine-tuning されても攻撃耐性があることが分かった。

4.3 プルーニング攻撃時の評価

モデル圧縮に対して本来のタスクの精度に影響がないかを調べた。図 6 はモデル圧縮を行い、0%から 5%ずつ増やして 95%まで行った時の accuracy と loss である。

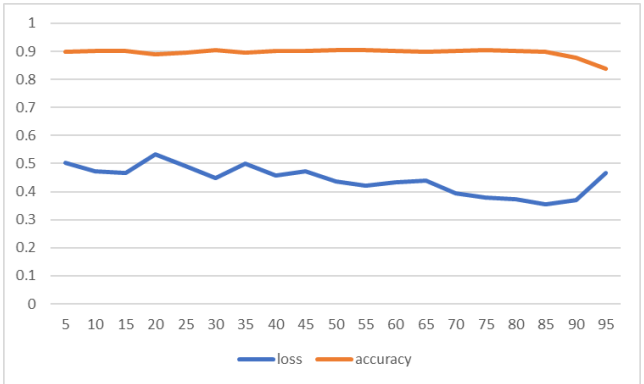

図 6 0%から 5%ずつ増やして 95%までモデル圧縮行った時の accuracy と loss

図 7 はモデル圧縮を行い、0%から 5%ずつ増やして 95%まで行った時のレイヤである。モデル圧縮は 4epoch で行った。こちらも電子透かしの攻撃耐性を調べるために calculate loss を使う。今回の実験での第 1 レイヤが layer1、第 2 レイヤが layer2 である。

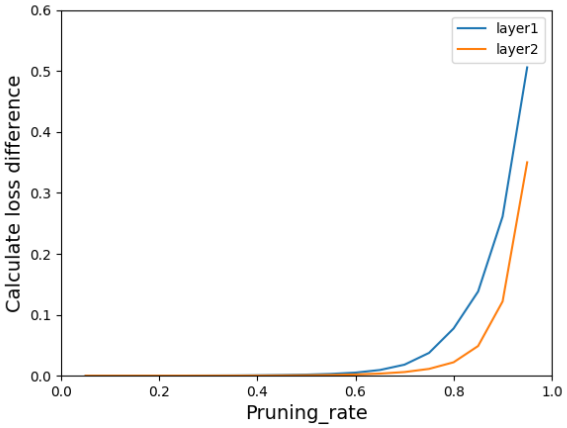

図 7 5%~95%のモデル圧縮時の calculate_loss 値からモデル圧縮無しの calculate_loss 値を引いた値

学習済みモデルに対して、モデル圧縮を行い、電子透かしに影響が出ないかの実験を行った。今回はモデル圧縮の 3 つの方法のうち Pruning を使った。Fine-tuning と同様に式(1)の calculate loss を使い、初期値からどれだけ変化しているかの差を比較した。モデル圧縮は 0%から 5%ずつ刻みで増やしていき、95%までの calculate loss を記録した。その値を 0%の calculate loss と比較することで、差が大きくなっているかどうかを調べた。

まず、図 7 のレイヤ 1 を見ると 70%以上になると著しく増加していることがわかる。70%の時のレイヤ 1 の重みを見ると以下の図 8 になった。

```
[[0.99903468 0.99937085 0.99822421 0.93829798 0.99871415 0.99995941
  0.99281892 0.99985369 0.99796757 0.9986427  0.99804556 0.9995978
  0.99488754 0.99946593 0.99952634 0.93979517 0.9882373  0.99780995
  0.99661345 0.99108123 0.99845155 0.99864285 0.99197104 0.99030756
  0.99678062 0.99843062 0.99845513 0.99869135 0.99909231 0.99937588
  0.99970169 0.99969767 0.9995191  0.91710536 0.99760245 0.99342663
  0.99879352 0.9931521  0.99987968 0.99974145 0.97473347 0.99989761
  0.99897789 0.99850443 0.99907706 0.99959903 0.99954106 0.96953041
  0.99919504 0.99713983 0.99980721 0.99964179 0.99962403 0.9989043
  0.99999226 0.98344165 0.99962623 0.98604192 0.99551164 0.99997068
  0.99580262 0.9927705  0.99692076 0.99932136 0.99312874 0.99996571
  0.99583573 0.99097679 0.99960827 0.98399441 0.99572268 0.85874561
  0.99977807 0.99845861 0.99984683 0.98279279 0.99949336 0.97234219
  0.99968046 0.99915221 0.99359354 0.98894549 0.99485427 0.99660407
  0.99976512 0.99779309 0.99969707 0.99621789 0.99954667 0.99201998
  0.9910546  0.99999688 0.99974248 0.99159314 0.99388879 0.99972924
  0.9992368  0.99890659 0.96759351 0.9908945  0.99248641 0.98399098
  0.99985748 0.99977429 0.99234331 0.9984413  0.99821614 0.98588587
  0.99836055 0.99992667 0.99788893 0.9857821  0.99989589 0.99791878
  0.98588303 0.99998002 0.99906076 0.99810268 0.99995205 0.99988368
  0.99943526 0.99945137 0.99190506 0.99739604 0.99282207 0.99801583
  0.99988841 0.9965262  0.98984437 0.9830327  0.99827594 0.99967868
  0.99996799 0.99978183 0.99916198 0.99472424 0.99976311 0.9892137
  0.99522693 0.99975478 0.99977987 0.99722922 0.96107597 0.99978492
  0.98411056 0.99139206 0.99965017 0.9940213  0.99999839 0.99998334
  0.99925368 0.99864523 0.97544476 0.98831789 0.99971473 0.96212706
  0.99417438 0.99922941 0.99987476 0.99202268 0.99969396 0.99962142
  0.99442153 0.99885775 0.99328425 0.99995844 0.99444548 0.99758019
  0.98770928 0.99999925 0.99911997 0.99941174 0.98846321 0.97956539
  0.99967845 0.994908   0.99999899 0.96702622 0.98747501 0.9587032
  0.99919346 0.99999978 0.99907784 0.99996893 0.99809193 0.99999513
  0.99705965 0.99848374 0.99890627 0.99748707 0.99364503 0.99694864
  0.99971883 0.99318988 0.99972846 0.99172916 0.99604971 0.98815765
  0.98507092 0.9968936  0.99483709 0.99573557 0.98006766 0.99535565
  0.99972731 0.98458577 0.98199238 0.99988789 0.99992896 0.98635818
  0.96900682 0.99992332 0.99498614 0.99882849 0.99897311 0.96614553
  0.99947677 0.99962793 0.99184578 0.99999999 0.99997337 0.98958753
  0.99999934 0.99999588 0.99534628 0.99892076 0.99858932 0.99923882
  0.98895513 0.99450959 0.9768503  0.99380039 0.99999432 0.99937322
  0.98749191 0.9969936  0.99610589 0.99941148 0.87239565 0.99327507
  0.99827322 0.9989336  0.99691021 0.99550982 0.99987886 0.99807184
  0.84209488 0.97447708 0.99794413 0.99223538 0.99272012 0.99886981
  0.9996995  0.98981007 0.99997751 0.99798949]]
```

図 8 モデル圧縮 70%のレイヤ 1 から検出した結果

図 8 では青い部分の値が明らかに変動しているため、電子透かしに悪影響を与えていることがわかる。

次に、図 7 のレイヤ 2 を見ると 75%以上になると増加傾向が高くなっている。75%の時のレイヤ 2 の重みを見ると以下の図 9 になった。

```
[[0.99892901 0.98190338 0.99475154 0.9979506  0.9776167  0.99133695
  0.99578846 0.99033153 0.99782716 0.99870853 0.99996929 0.99860843
  0.99634995 0.99466085 0.99709244 0.99700688 0.98458785 0.9948812
  0.99829886 0.99928343 0.98440547 0.99376069 0.99754804 0.99798495
  0.99820431 0.99854745 0.99759391 0.99627854 0.99942449 0.99396991
  0.99535325 0.99787051 0.99365721 0.99781711 0.9969364  0.99792106
  0.99490723 0.97093732 0.99449388 0.98853145 0.9962089  0.97876375
  0.99745943 0.99746291 0.99801961 0.98925185 0.97593368 0.99697726
  0.99812988 0.99968275 0.99665744 0.98986931 0.99841525 0.99669749
  0.99983322 0.99729713 0.99707136 0.99987101 0.99261981 0.99919757
  0.99901159 0.99015925 0.99917284 0.99719637 0.99791495 0.99555822
  0.9940397  0.99539192 0.99674021 0.99770496 0.99581579 0.99733136
  0.99932312 0.99477811 0.99903847 0.99711224 0.99898746 0.99816026
  0.99655935 0.99917962 0.99742654 0.99506346 0.99926496 0.99727316
  0.99689225 0.99878586 0.98711508 0.99737792 0.98918668 0.99935066
  0.99859231 0.99865258 0.99890347 0.99875768 0.99807806 0.99818891
  0.99502083 0.99631251 0.99724961 0.99358534 0.9666587  0.9977799
  0.99861408 0.99769125 0.99717039 0.99828056 0.99622513 0.99733196
  0.99778998 0.99024865 0.99895097 0.98830038 0.99829651 0.99806397
  0.99736671 0.99665588 0.99642658 0.99688568 0.99426298 0.99661411
  0.99536398 0.99655385 0.99910925 0.99951381 0.99355867 0.99704802
  0.99956562 0.99801589 0.96703976 0.99685464 0.99789293 0.99549778
  0.99779156 0.99826303 0.99886581 0.99891917 0.99725848 0.99320611
  0.99242105 0.99680312 0.99596812 0.99475861 0.9970108  0.99833428
  0.99554718 0.99454814 0.98935112 0.99820222 0.9983932  0.99769819
  0.9861522  0.9971382  0.99606825 0.99411139 0.99792515 0.99923228
  0.9959398  0.99601933 0.997406   0.99693082 0.99787278 0.99470972
  0.99218884 0.98765659 0.99932691 0.99294715 0.99899974 0.99538501
  0.99889987 0.99847686 0.98975471 0.99980537 0.99783813 0.99702378
  0.99881371 0.99384214 0.99717774 0.99838306 0.99893116 0.9907703
  0.99860341 0.98787022 0.99930176 0.99438089 0.99448612 0.99432154
  0.99244416 0.99532348 0.98195171 0.98990634 0.9986731  0.99611039
  0.99219913 0.9972289  0.9857253  0.99814809 0.9954507  0.99042305
  0.99305767 0.9929473  0.99650747 0.99893621 0.99729481 0.99612402
  0.99892002 0.99651909 0.98572338 0.97478551 0.99590307 0.9993968
  0.99050842 0.98790226 0.99650668 0.99217109 0.99775925 0.99782974
  0.9982033  0.99613948 0.99631629 0.98705858 0.99714853 0.99467617
  0.99879465 0.99671944 0.99624953 0.9946351  0.99895363 0.99915746
  0.99694546 0.99901893 0.99489015 0.99979425 0.99561171 0.99502957
  0.99932052 0.99287953 0.99740845 0.98766508 0.99936642 0.99950103
  0.99342026 0.99586475 0.99723243 0.99786743 0.99349482 0.99885896
  0.99849928 0.99787072 0.99357325 0.99572263 0.98517745 0.98918707
  0.99717126 0.99575562 0.99803417 0.99332559]]
```

図 9　モデル圧縮 75%のレイヤ 2 から検出した結果

図 9 では図 8 のような明らかに変わっているような値は見えなかった。しかし、モデル圧縮の前後での calculate loss を比較すると以下の表 5 のようになり、電子透かしへの影響が出ていることが分かる。

表 5 モデル圧縮(75%)前後のレイヤ 2 の calculate loss

	レイヤー2
モデル圧縮(75%)前	0.00032884
モデル圧縮(75%)後	0.004765931
モデル圧縮との比較（後/前）	14.49317614

5. むすび

今回は、ディープラーニングモデルの著作権保護のために電子透かしを適用した。既存技術を元に電子透かしの埋め込みレイヤを追加して、埋め込みビット数を 256bit から 512bit に拡張して、評価を行った。データセットは Cifar-10、モデルは WRN(Wide-Residual-Network)を使用して、攻撃耐性として Fine-tuning とモデル圧縮を行った。結果としては、電子透かしはしっかりと入っており、本来の画像認識タスクの精度では Test error に少しだけ影響受けたが、精度は高いままを維持している。学習済みモデルに対して Fine-tuning を行っても、影響を受けることはほとんどなく、モデル圧縮では電子透かしへの影響は観測されるものの、ビット誤りにまでは至っていない。

謝辞　本研究は JSPS 科研費 JP18K11309 の助成を受けたものです。

参考文献

[1] 知的財産戦略本部 『新たな情報財検討委員会　新たな情報財検討委員会報告書』 (2017)

[2] Uchida Yusuke, et al. "Embedding watermarks into deep neural networks." *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. ACM, 2017.

[3] Cifar-10 https://www.cs.toronto.edu/~kriz/cifar.html
2020/2/5 アクセス