

理 工 学 部

2020年3月

卒 業 論 文

機械学習を用いたファズデータの
チェックサム及びハッシュ値の推定

27016627 藤 本 高 史

(情 報 科 学 科)

機械学習を用いたファズデータの チェックサム及びハッシュ値の推定

藤本 高史

内容梗概

本研究では、機械学習を用いてデータのチェックサム及びハッシュ値の推定を行う手法を提案する。正当又は不当なデータをソフトウェアに入力することによって、ソフトウェアの脆弱性のテストを行う変異ベースのファジング手法では、チェックサムやハッシュ値により入力データは殆ど通過しないため、網羅率が低いという課題があげられる。これに対し、難波はニューラルネットワークによるチェックサムの推定を行ったが、入力データが 8 byte の固定長しか学習を行っていないため、汎用性に欠ける。これを解決するため、本研究では 8 byte 以上の入力データに対するチェックサム及びハッシュ値の推定を行う。機械学習を行うための学習モデルに Encoder・Decoder モデルを使用する。Keras を使用してニューラルネットワークの構築および学習を行った。結果、チェックサムはランダム文字列に対して 20%、英文に対して 50.8%、CRC16 はランダム文字列に対して 9.01%、英文に対して 50.8%、CRC32 はランダム文字列に対して 11.9%、英文に対して 4.17% の正答率が得られた。

キーワード

機械学習, ニューラルネットワーク, ファジング, Encoder・Decoder, 変異ベース, チェックサム, ハッシュ

目次

第1章 序論	1
第2章 ニューラルネットワークを用いた チェックサムの推定	3
2.1 チェックサムとCRC 及びハッシュ値	3
2.2 ニューラルネットワークを用いたチェックサムの推定	4
第3章 機械学習によるチェックサム及び ハッシュ値の推定	9
3.1 可変長データに対応したニューラルネットワークによる学習	9
3.2 Encoder・Decoder モデルを用いた学習	11
3.2.1 Encoder・Decoder モデル	11
3.2.2 ニューラルネットワークの構成	11
第4章 実装と実験結果	15
4.1 実装	15
4.2 実験	15
4.3 学習結果	16
4.4 学習結果の検証	17
4.5 考察	21
第5章 結論	23
謝辞	25
参考文献	27

第1章 序論

近年, 様々なシステムやソフトウェアのプログラムに対するセキュリティは大きな注目を浴びている. プログラムやシステムの脆弱性が存在すると, 不具合が生じたり, 第三者からの攻撃によって様々な被害が出る恐れがある. これらの被害を防ぐため, プログラムやシステムには徹底的なテストが必然となっている.

その脆弱性を検出するテストの一手法にファジングと呼ばれる手法がある. 与えられたプログラムに対して, 本来想定されていない情報を大量に入力し, その入力に対する動作に異常がないかどうかをテストする.

ファジングで使用するデータの生成方法として変異ベース手法がある. 変異ベース手法は既存の入力データの文字列等に該当する部分を変化させて新たなテストデータを生成する手法である. 入力データの一部を変異させるため, テストデータの作成に時間はかからず汎用性も高いが, テストデータの質は低い. 特に, テスト対象のプログラムにデータ検査としてチェックサムやハッシュ値が存在すると, 破損データとして扱われ, 入力データ通過率が非常に低くなるという課題がある.

その課題を解決するために, 難波 [1] は, ニューラルネットワークを用いて入力データに対するチェックサムの学習及び推定を行い, 68%の正答率が得られた. しかし, この学習済ニューラルネットワークは, 8 byte の固定長のデータに対するチェックサムのみを対応していないため, ファジングツールへ実装をすると, 実際に使用することができる場面がかなり限定的になってしまうので, 汎用性に欠けるという課題がある.

本研究では難波 [1] の汎用性の向上を目的に, 8 byte 以上の入力データに対するチェックサム及びハッシュ値の推定を行う. 推定にニューラルネットワークを使用し, 特に (Seq2Seq)Encoder・Decoder モデル [2] を用いる. 入力データは可変長データで学習を行う. 推定する値としてチェックサム, CRC16[3], CRC32[3], MD5[4], SHA1[5] を採用する. 結果として, チェックサム及び CRC16[3], CRC32[3] に対して高い精度の正答率が得られた.

以下, 2 章で難波 [1] の内容と課題について説明し, 3 章でニューラルネットワークを用いたの課題克服方法について述べ, 4 章で実験内容と実験結果を述べ, 5 章で結論とする.


第2章 ニューラルネットワークを用いた チェックサムの推定

2.1 チェックサムと CRC 及びハッシュ値

チェックサムは誤り検出符号の一種であり、特定の計算方法を用いて元になるデータから一定の計算手順により求められた、規則性のない値のことである。主にデータの検査をするために用いられる。

データの検査を行うとき、入力するデータ全体や分割したデータに対して計算を行い、その結果をデータに付加する。そして、チェックサム及びハッシュ値を計算した算術と同じ方法で再度計算を行い、その値がデータの後ろに付加されている値と比較して一致するかどうか確認を行う。一致していなければ、検査したデータは破損したデータとして扱われる。

チェックサムの計算方法は、対象となる文字列に対して各文字を全て ASCII コードに置き換え、それを合算した値から 256 の剰余をとる。計算方法の例を図 2.1 に示す。図にある Thank you very much. という文字列のチェックサムを計算を行うとすると、初めに各文字全てを ASCII コードに準拠した整数に置き換える。そして、置き換えた値を全て加算した値から 256 の剰余を行うことでチェックサムが得られる。最後に、文字列に計算したチェックサムを付加する。

Thank you very much.  $28 + 88 + 129 + \dots + 223 = 12985$

$$12985 \bmod 256 = 128$$

Thank you very much. 128

図 2.1: チェックサムの計算例

ハッシュ値は、ハッシュ関数から出力された規則性のない固定長のビット列である。ハッシュ値の計算方法はハッシュ関数によって様々である。ハッシュ値は一方関数なため、ハッシュ値から元の文字列を特定することはできない。その性質を利用して、暗号などのセキュリティの分野で多く使用されている。

CRC[3] は誤り検出符号の一種であり、データから特定の定数の剰余を検査用の値として用いる。主に PNG や ZIP などのファイルのデータ検査や、HDLC 手順や CSMA/CD 方式などの誤りチェック及びノイズチェック使われている。出力される bit 数は CRC[3] の種類によって異なる。例えば、

CRC16[3] であれば 16 bit 以下の 2 進数が出力され、CRC32[3] では 32 bit 以下の 2 進数が出力される。

MD5[4] 及び SHA1[5] は暗号学的ハッシュ関数の一種であり、MD5[4] はデータを元に出した 128 bit の値、SHA1[5] は 160 bit の値を用いる。出力される bit 数は、どんなデータでも必ず定められた固定長で出力される。IPsec, SMTP などのセキュリティープロトコルで使用されている他、暗号や認証、デジタル署名などにも用いられている。

CRC[3], MD5[4], SHA1[5] の出力値の例を図 2.2 に示す。Hello という文字列から CRC16[3] を計算すると、1 と 0 の 2 進数の値が出力される。出力される bit 数が 16 bit 以下となるのは、CRC は特定の方程式から文字列を 2 進数に表した値の剰余を出力とするためである。Good という文字列から MD5[4], Nice という文字列から SHA1[5] を計算すると、0 ~ F の 16 進数の値が出力される。出力される bit 数は、必ず固定長である。



図 2.2: CRC, MD5, SHA1 のハッシュ値

2.2 ニューラルネットワークを用いたチェックサムの推定

ニューラルネットワークはパーセプトロンのアルゴリズムを 3 層にし、誤差逆伝播法と呼ばれる学習機能を持たせた多層パーセプトロンである。一般的なニューラルネットワークを図 2.2 に示す。ニューラルネットワークは入力層、中間層、出力層の 3 つの層で構成されている。入力層は情報を入力するためのニューロンであり、入力層から情報を受け取った中間層は、情報を分析し、学習した後にその結果を次の層のニューロンに繰り返し伝達させ、最終的に出力層で結果が出力される。そこで、ニューラルネットワークが予測した値と真理値を比較し、誤差逆伝播により最適な重みへと更新していく。

RNN(Recurrent Neural Network) は、ニューラルネットワークの各層の出力を再度、その層の出力にそのまま再帰するという特徴をもつニューラルネットワークである。RNN の構成を図 2.2 を以下に示す。主に時系列データと呼ばれる時間的な順序に従って一定の感覚で集められたデータを分析、計算するという役割をもつ。層の構成として、入力層、中間層、出力層となっているが、中

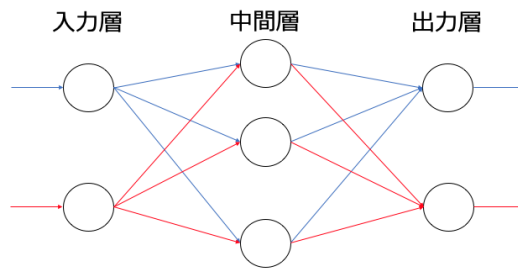


図 2.3: ニューラルネットワーク

間層は過去の情報を保持する。しかし、RNN は過去の情報を全て記憶した状態で計算を行うので、計算量が多くなると共に、計算に不要な情報までも取り入れてしまう欠点をもつ。

この欠点を解決したのが LSTM である。LSTM の構成を図 2.5 に示す。中間層に入力ゲート、忘却ゲート、出力ゲートの 3 つのゲートを利用して計算を行う。この 3 つのゲートはそれぞれ、計算を行う上で何が必要で何が不必要なのか取捨選択をすることができる。

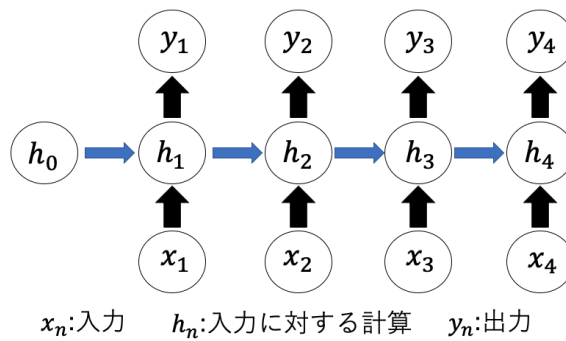


図 2.4: RNN

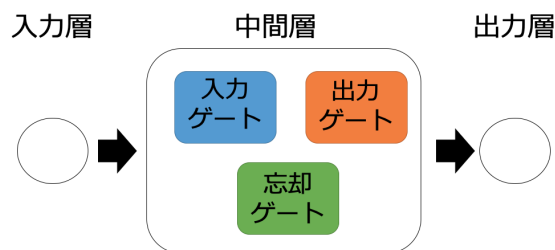


図 2.5: LSTM

難波 [1] はニューラルネットワークを用いて、8 byte の入力データに対するチェックサムの推定を行っている。正規データからデータとチェックサムの集合を抽出し、ニューラルネットワークに学習させる。学習済ニューラルネットワークで変異データからチェックサムを推定し、更新する。学習の仕組みを、図 2.7 に示す。初めに、文字列とその文字列に対するチェックサム及びハッシュ値の

2 種類のデータを用意する。次に、そのデータをニューラルネットワークへ入力をし、学習を行う。そして、学習済ニューラルネットワークに対して、文字列を入力することにより、チェックサム及びハッシュ値を推定する。最後に、推定したチェックサム及びハッシュ値を、文字列の末端に付与する。よって、正当な入力データとして扱われる。

難波 [1] のニューラルネットワークの構造を図 2.6 に示す。各層の数字はノード数を表す。初めにランダム文字列から 1 つ 1 つの文字を分割して入力を行い、Embedding 層で入力データのベクトル化を行う。次に Embedding 層から受け取った情報を元に LSTM 層で計算を行い、ドロップアウトで過学習を防ぎつつ、全結合層を通してから、最後に出力層で 256 段階の評価を行うことで、チェックサムの推定を行う。このように学習を行ったニューラルネットワークを使って、チェックサムを生成するデータのみを入力し、推定したチェックサムを出力する。最後に、出力されたチェックサムと生成データを合成する。これにより、チェックサム部分をニューラルネットワークによって推定した物に入れ替えた新しい入力データが生成されるという仕組みである。

難波 [1] の課題は、8 byte の固定長の入力データに対する学習しかさせていないことである。実際にこの学習済 NN を実装しようとなると、入力データの文字列が 8 byte またはそれ以下でないと正しいチェックサムを推定することが難しいと考えられる。また、学習はチェックサムのみを行っていないため、もし入力データにハッシュ値が採用されているとなると、推定を行うことができない。ゆえに、汎用性が低いことがあげられる。

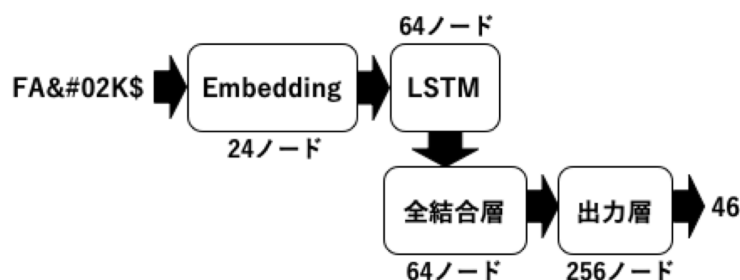


図 2.6: 難波 [1] のニューラルネットワーク

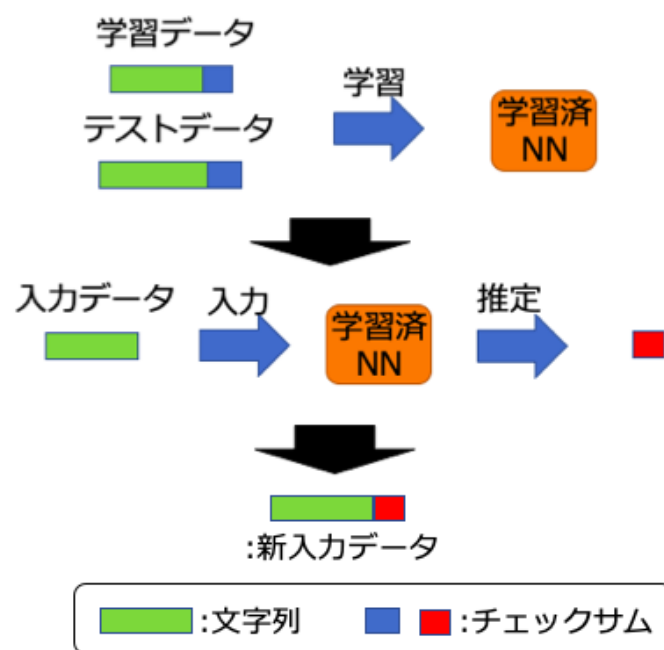


図 2.7: チェックサムの推定

第3章 機械学習によるチェックサム及びハッシュ値の推定

3.1 可変長データに対応したニューラルネットワークによる学習

本論文では、難波 [1] の汎用性を高めるため、8 byte 以上のデータに対するチェックサム及び様々なハッシュ値の推定を行う手法を提案する。チェックサムおよびハッシュ値の学習には、文字列とその文字列に対するチェックサムおよびハッシュ値のデータセットを使用する。データとチェックサムおよびハッシュ値の位置とハッシュ関数の種類は既知であることを前提とする。チェックサム及びハッシュ値の学習にはニューラルネットワークを使用して行う。

学習の仕組みを図 3.1 に示す。学習の仕組みは難波 [1] と同様である。難波 [1] と異なる点として、チェックサム以外に CRC[3], MD5[4], SHA1[5] のハッシュ値を推定する。また、[1] byte の固定長の学習しか行なっていなかったため、様々な長さの文字列である可変長データで学習を行う。

学習に使用するニューラルネットワークは、Encoder・Decoder モデル [2] を利用したニューラルネットワークを構成して学習を行う。また、ハッシュ関数によって計算方法は変わってくるため、各々のハッシュ関数に対応した機械学習を行わなければならない。

可変長データの例を図 3.2 に示す。記号や数字、アルファベットの文字を 1 byte として扱う。学習を行うデータとして、ランダム及び規則性のある文字列を扱う。

文字列の例を図 3.3 と図 3.4 に示す。ランダム文字列は、数字、記号、アルファベットを乱雑に並べた文字列である。この文字列を学習させることで、バイナリデータを変異ベース手法で入力データを作成するときに役立つ。規則性のある文字列は、日本語や英文など、単語間に関係性をもつ文字列である。この文字列を学習させることで、主にテキストデータなどを変異ベース手法で入力データを作成するときに役立つ。

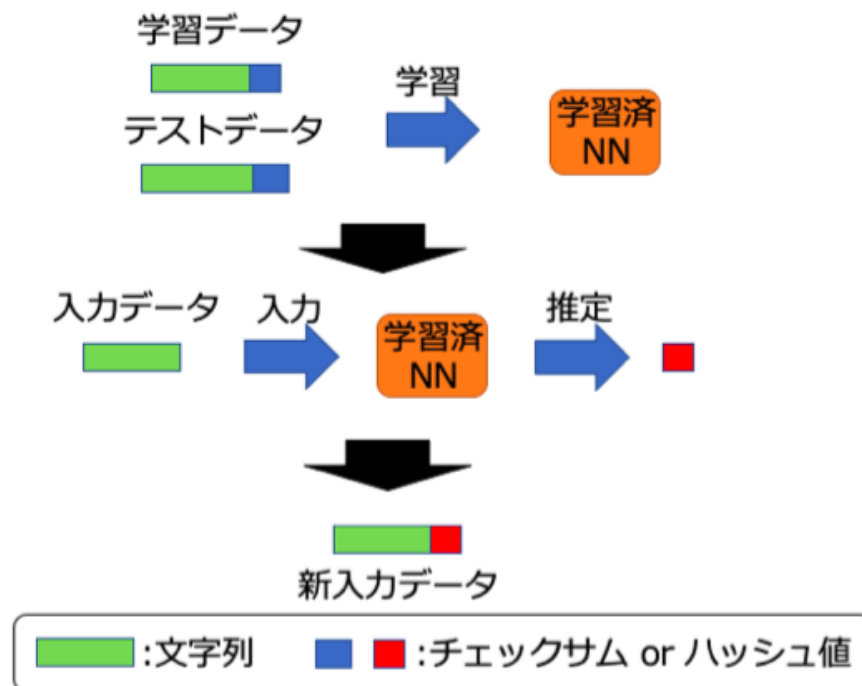


図 3.1: チェックサム及びハッシュ値の推定

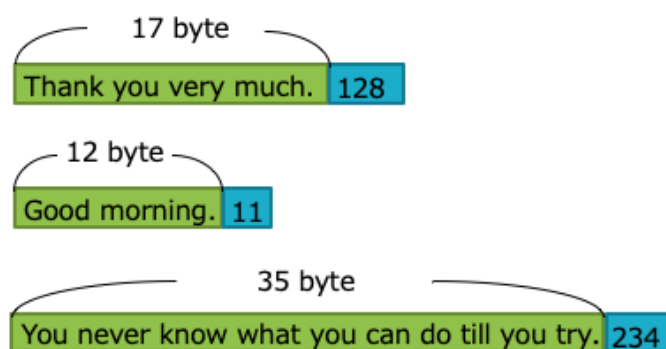


図 3.2: 可変長データ

galwry3ial2lau3Wd 55

D21jhg("#mf~~nk~~iea" 27

R#(Yhc73aLKYWCA 211

図 3.3: ランダム文字列

Thank you very much. 128

How are you ? 189

I am able to fly in sky 4

図 3.4: 規則性のある文字列

3.2 Encoder・Decoder モデルを用いた学習

3.2.1 Encoder・Decoder モデル

Encoder・Decoder モデル [1] は自然言語処理系のニューラルネットワークである。主に翻訳や文章の要約、対話生成に使用される。

モデルの仕組みを図 3.5 に示す。この図は、英語から日本語へと翻訳を行うニューラルネットワークである。例として、Thank you very much. という文字列を Encoder へ入力する。

Encoder は画像、テキストなど何かしらの特徴ベクトルに変換する機構のことである。入力した文字列に対して学習を行い、その処理した結果として内部状態を返す。内部状態は、入力データに対してベクトル化の処理を行った出力である。ベクトル化を行った際、各単語のベクトルの長さは全て一定にしなければならない。次に、ベクトル化をした各単語の特徴ベクトルを Decoder へ入力する。

Decoder は Encoder でエンコードされた特徴ベクトルをデコードして何か新しいデータを生む機構のことである。ターゲットとなる文字列の前の文字が与えられた場合、その次の文字を予測するように学習を行う。学習を行う際、Decoder への入力を Encoder へ入力した文字列の答えとなる文字列をあらかじめ用意する。この時、出力されるデータは、Encoder で入力したデータと同じデータ形式である必要はなく、音声や動画など様々な形式で出力が可能である。

3.2.2 ニューラルネットワークの構成

ニューラルネットワークの構成は図 3.6 を用いる。Encoder にあたる箇所は、input_1 から lstm_1 までの層となる。input_1 は文字列を入れる入力層である。図の (None, None) はタプルであり、主に層のノード数を表す。各層の input は、層へ入力されるタプルで、output は次の層へ出力するタプルである。None の意味は、任意の正の整数を意味する。入力文字列の長さは一定ではないため、入力層のノード数は各入力文字列の長さとなる。

embedding_1 は input_1 から受け取った文字列のベクトル化を行う。図の output の 512 がノード数であり、ベクトル化された文字の配列の長さとなる。

lstm_1 は embedding_1 から受け取った情報を元に LSTM によって計算を行う。この時、計算結果は破棄し、計算過程などの内部状態のみを返す。

Decoder にあたる箇所は、input_2 から dense_1 までの層となる。input_2 は Encoder へ入力した文字列に対する出力となる文字列を入れる入力層である。input_1 と同様、ノード数は一定ではない。

embedding_2 は embedding_1 と同様にベクトル化を行い、lstm_2 で計算を行う。

lstm_2 では、lstm_1 と違い、内部状態を破棄し、計算結果を返す。最終的に、lstm_2 で計算された結果を dense_1 の出力層で結果を出力する。出力層のノード数が 3 となっているのは、出力する文字数が 3 文字であることを意味する。

各ノード数などのパラメータは、どのような入力及び出力を行うかによって最適な値が異なるため、何度も実験を繰り返して最適な値を見つける必要がある。また、全てのチェックサム及びハッシュ値を学習するにおいて、ニューラルネットワークの構成は同じである。

出力層は、一般的なニューラルネットワークと違い、出力される各文字に対して振り分けられる。例として、図 3.7 にしめす。Thank you!という文字列から 180 となるチェックサムを推定するならば、'Thank', 'you', '!' の 3 つの単語に分割し、入力をして学習を行う。推定を行う際に、初めの 1 文字目が 1 ~ 9 または空白なのかを 1/10 で予測し、出力する。それを最後の文字まで続け、その結果の予測値と真実値が同じになるように重みを更新していく。I don't think so. という文字列の推定結果の 1 文字目の < > は空白を表す。この空白は、1 文字として扱う。3 つ目の I am fine. から CRC16 を推定するならば、0 と 1 及び空白の 1/3 で各文字を予測する。



図 3.5: Encoder・Decoder モデル

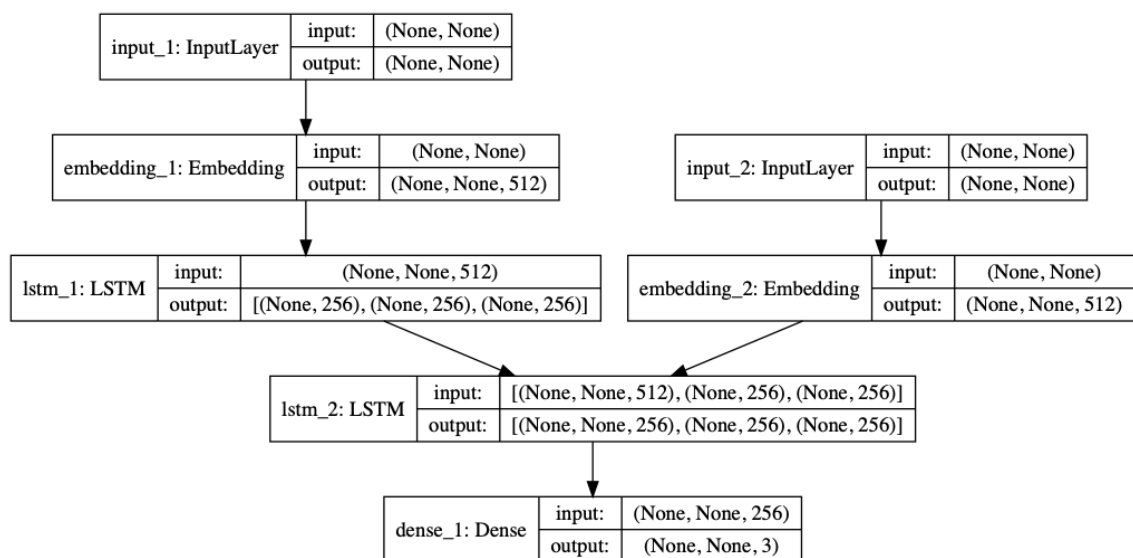


図 3.6: ニューラルネットワーク

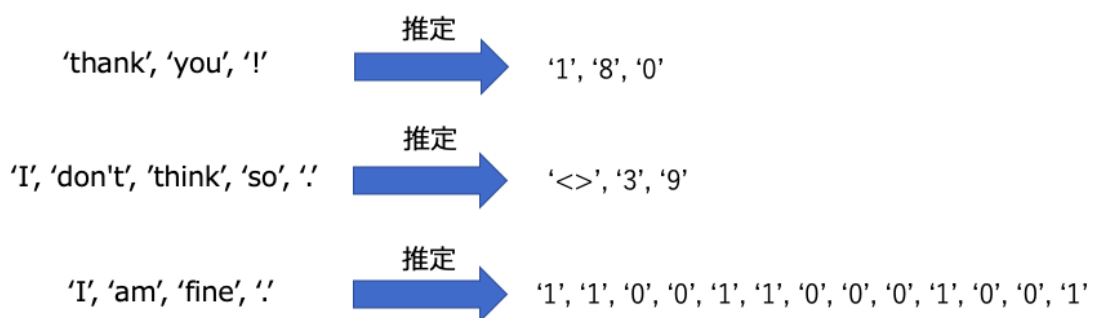


図 3.7: 出力の構成

第4章 実装と実験結果

4.1 実装

本研究では、機械学習を用いて 8 byte 以上のランダム文字列と規則性のある文字列に対するチェックサム及びハッシュ値の推定を行った。データセットは、Perl にある文字列からチェックサム及びハッシュ値を計算するモジュールを使用して作成した。文字列には、ランダム文字列と規則性のある文字列として英文を起用した。提案手法を Keras を用いて Python で実装し、機械学習を実行した。文字列とチェックサム及びハッシュ値のデータセットは、各学習に置いて最適なデータ数を用意した。ハッシュ関数は CRC16[3], CRC32[3], MD5[4], SHA1[5] を使用した。学習に用いたニューラルネットワークは、Encoder・Decoder モデルを使用した。学習回数はチェックサム及びハッシュ値によって異なる。オプティマイザには Adam を使用した。

4.2 実験

機械学習を用いてランダム文字列と規則性のある文字列に対するチェックサム及びハッシュ値を推定する実験を行った。学習の仕組みを図 4.1 に示す。初めに、文字列と文字列に対するチェックサム及びハッシュ値の 2 種類のデータを用意する。次に、そのデータを教師データとテストデータに分割する。教師データは、ニューラルネットワークに学習させるためのデータであり、テストデータは、実際に学習ができているのか確認するためのデータである。そして、教師データを使って学習を行う。教師データ全てを使って一度学習を終えたのちに、テストデータで確認を行う。ニューラルネットワークが予測した値と正解の値を比較し、間違っていれば、その乖離をなくすために再度学習を行う。この時、汎化性能を高めるために、学習精度は教師データで学ばせた時より、テストデータで確認を行った時の方が高いように学習を行わなければならない。

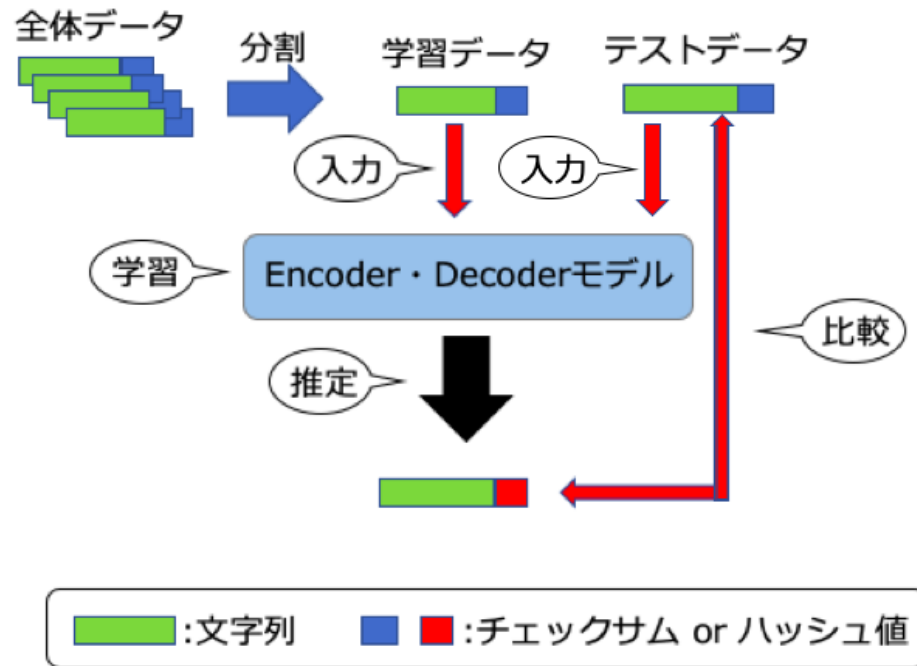


図 4.1: 学習の流れ

4.3 学習結果

英文の学習結果を、表 4.1 に示す。文字列は可変長の入力データである。train data と test data は、学習データとテストデータで学習を行った時の学習精度を表す。train loss 及び test loss は、学習及び推定した結果と実際の正解と比較して、どれほどの乖離があったのかを表す。byte は、学習を行った文字列の byte 数を表す。epoch は、学習を行った回数を表す。lstm node は、LSTM のノード数を表す。

全体のテストデータの結果から、チェックサムと各ハッシュにおいて、学習をしていない状態より遥かに高い精度が得られた。また、学習データとテストデータがほぼ同じ精度となっているため、理想的な機械学習をしていると考えられる。しかし、MD5 と SHA1 の損失値が高い傾向にあるため、学習方法を改善しなければならない。

ランダム文字列の学習結果を、表 4.2 に示す。文字列は可変長の入力データである。train data と test data は、各データで学習を行った時の学習精度を表す。train loss 及び test loss は、学習及び推定した結果と実際の正解と比較して、どれほどの乖離があったのかを表す。表の byte という項目は、学習を行った文字列の byte 数を表す。epoch は、学習を行った回数を表す。

規則性のある文字列と比較を行うと、全体的に精度が低い傾向となっている。また、学習誤差、推定誤差も高い傾向にある。原因として、文字列に関係性がないため、そこから正解を予測するための基準を理解させるのが困難であると考えられる。

表 4.1: 英文の学習精度

	train data	test data	train loss	test loss	byte	epoch	data	lstm node
checksum	50%	50%	0.27	0.24	2 ~ 48	100	12 万	64
CRC16	47%	47%	0.57	0.57	2 ~ 16	120	12 万	64
CRC32	49%	50%	0.59	0.59	2 ~ 18	100	10 万	48
MD5	11%	11%	2.45	2.45	2 ~ 19	100	20 万	128
SHA1	19%	19%	2.2	2.2	2 ~ 18	100	12 万	256

表 4.2: ランダム文字列の学習精度

	train data	test data	train loss	test loss	byte	epoch	data	lstm node
checksum	51%	52%	0.48	0.48	1 ~ 32	300	10 万	64
CRC16	53%	53%	0.53	0.53	1 ~ 16	130	16 万	32
CRC32	54%	54%	0.57	0.57	1 ~ 8	100	10 万	52
MD5	11%	11%	2.27	2.22	1 ~ 8	100	20 万	128
SHA1	14%	14%	2.47	2.47	1 ~ 8	50	20 万	128

4.4 学習結果の検証

表 4.1 と表 4.2 に示した結果は期待以上の成果であったが、これらの学習精度は目安としかならない。一般的なニューラルネットワークの場合、学習精度はニューラルネットワークが予測した結果と実際の正解と完全に一致した数から、検査したデータ数の割合から出力される。この Encoder・Decoder モデル [2] は、出力した結果の 1 文字ずつに対して判定を行う。つまり、予測した結果と実際の正解が完全に一致しなくても、正解の文字が 1 つでも含まれていれば良い。そのため、実際に推定を行う際と異なる正答率である可能性が高いことがわかる。

本来の学習精度を測るために、学習済ニューラルネットワークを使って、文字列からチェックサム及びハッシュ値の推定を行った。ランダム文字列及び英文の正答率を表 4.3 に示す。random, english が学習済ニューラルネットワークを使用した時に得られた正答率である。train random, train english は、学習時に得られた正答率である。検証に使用したデータは、学習時に使用したテストデータにある 1 万個の文書を用いて行なった。

学習時に得られた正答率と学習済ニューラルネットワークを使用して得られた正答率を比較すると、必ずしも正答率が一致しないことがわかった。特に、チェックサムまたはハッシュ値によって、ランダム文字列の方が正答率が高かったり、英文の方が正答率が高かったりなど、ばらつきが見られた。原因として、適切な LSTM のノード数を設定出来ていなかったり、データセットが不十分であったりなど考えられる。また、ニューラルネットワークの構成自体が適切でない可能性もあるため、更なる実験及び検証が必要である。

表 4.3: 学習済ニューラルネットワークを用いたチェックサム及びハッシュ値の推定

	random	english	train random	train english
checksum	20%	50.8%	52%	50%
CRC16	9.01%	9.08%	53%	47%
CRC32	11.9%	4.17%	54%	50%
MD5	12.07%	2.53%	11%	11%
SHA1	4.95%	10.75%	14%	19%

1 ～ 32 byte のランダム文字列と 2 ～ 48 byte の英文に対するチェックサムの推定結果の一部を表 4.4 に示す. Input が入力文字列であり, Decode が推定結果, Accuracy が入力文字列に対する正解の出力である.

表 4.4: チェックサムの推定

	Input	Decode	Accuracy
Random	6duf]LiT}‘KE:UCJ	110	100
	773=d#E4k%1UF2q	0	0
	8H?v;+6	208	209
English	may i profess my love?	14	13
	i’m telling you that...	208	208
	get him down.	165	164

推定結果から, 不正解だった推定結果と正解を比較すると, 正解から 1 ～ 10 の誤差で値を出力していることがわかった. 原因は調査中であるが, 一部の文字列に対して計算を行っていない可能性がある. この原因を解決することが出来れば, 更なる精度向上が見込まれる. ランダム文字列の結果と比較すると, 英文などの規則性のある文字列の方が高い精度が得られることがわかる. 原因として, 規則性のある文字列の方が, 文字列の関係性がはっきりしているため, 計算アルゴリズムが理解しやすいと考えられる.

1 ～ 16 byte のランダム文字列と 2 ～ 16 byte の英文に対する CRC16 の推定結果の一部を表 4.5 に, 1 ～ 8 byte のランダム文字列と 2 ～ 18 byte の英文に対する CRC32 の推定結果の一部を表 4.6 に, 示す. Input が入力文字列であり, Decode が推定結果, Accuracy が入力文字列に対する正解の出力である.

表 4.5: CRC16 の推定

	Input	Decode	Accuracy
Random	b%.H] [0@B!âP	110101010101	101010101110
	EER9V{Ta	11100101	11100101
	wYACk1LUX5fW	110101010110	101010101010
English	i do.	1111101010111011	1111101010111011
	just do it.	1010100010100101	10100000111011
	so i was happy.	1011011111110101	101000101101001

表 4.6: CRC32 の推定

	Input	Decode	Accuracy
Random	muah	0010101001110011000000111111101	1001000011101001010001101101
	o	1111000011111001001101000100	1111000011111001001101000100
	=!#U^	1110010111110011100011110011010	11010010010101110101101011100011
English	storm!	10011000001011001100011000011100	10011000001011001100011000011100
	stop! listen!	1111110000111111100010111111111	10110110010101100111010111111000
	who's a killer?	1111010000111111100010000001111	1000100110110000001000100111001

CRC16 の推定結果から, 正解に近い値を出力していることが多く, 完全に答えと一致しているものは少ないことがわかった. また, 正解と推定結果の bit 数が殆ど一致しているのが見受けられた. 英文と比較して, ランダム文字列と同様, 不正解だった出力でも正解と近い出力をしていることがわかった. しかし, より長い文字列の場合, 殆ど推定が出来ていなかった.

CRC32 の推定結果から, CRC16 と同様に, 正解に近い値を出力しているが, 完全に答えと一致している値は少なかった. また, 出力が全て固定長でなく, 様々な組み合わせの出力がされていることが見受けられるため, 学習が全くできていないという可能性は低いことがわかった. しかし, 正解と同じ値が出力された文字列の多くが 1 または 2 byte の時だった. 学習精度から CRC16 と比較して, CRC32 の方が正確に出力していることがわかった. 出力が短い方が推定するのが容易であると考えていたが, 恐らく学習させた文字列の長さの違いによるものである可能性が高い.

1 ～ 8 byte のランダム文字列と 2 ～ 19 byte の英文に対する MD5 の推定結果の一部を表 4.7 に示す. Input が入力文字列であり, Decode が推定結果, Accuracy が入力文字列に対する正解の出力である.

表 4.7: MD5 の推定

	Input	Decode	Accuracy
Random	POB3}-{g	b7bc8c67d658878888888946646664b6	400577e173ace391fc109e78f737d58d
	y:n}W,/S	5dc4d6466666666668888888888883877	b2d6d53f53a22a4117c97f92da16bfda
	H	c1d9f50f86825a1a2302ec2449c17196	c1d9f50f86825a1a2302ec2449c17196
English	what!?	7ee3f61af1b2b7f3f1607aac1cc86f8c	7ee3f61af1b2b7f3f1607aac1cc86f8c
	it is?	95fa3ec416cbdf2adf2a6ae4bca3435f	95fa3ec416cbdf2adf2a6ae4bca3435f
	mumbler!	b040dd95a762d45aab4f4f54f4f33592	3b7642c8f4a0a25539dff4aaeb0509cf

ランダム文字列は, どの入力文字列に対しても, 同じような出力をしていることがわかる. 特定の文字や文字列の長さに関わらず出力しているため, 原因はわからなかった. また, 実際に推定ができた文字列は, 入力文字列が 1 byte のみであった, このことから, MD5 のアルゴリズムをニューラルネットワークで理解させるのは非常に困難であることがわかった. 英文は, 6 byte 以下の文字列であれば, 正解又は正解に近い値を出力することがわかった. これより, ランダム文字列よりの正確に推定結果を出力出来ていることがわかった. ランダム文字列の方が正答率が高いが, 正解を出力することができる文字列は英文の方が高かった.

1 ～ 8 byte のランダム文字列と 2 ～ 18 byte の英文に対する SHA1 の推定結果の一部を表 4.8 に示す. Input が入力文字列であり, Decode が推定結果, Accuracy が入力文字列に対する正解の出力である.

表 4.8: SHA1 の推定

	Input	Decode	Accuracy
Random	(28ed3a797da3c48c309a4ef792147f3c56cfec40	28ed3a797da3c48c309a4ef792147f3c56cfec40
	pz	e0fd60007dd0003a70071010701060606d64bbbb536	5ffaa7bd95eb19342dcbb20cc58fc75e712c5847
	vndf3	1000710003a64607070707011d0070707070fc6fc	2fcc715443472d5bc62bde7300e7ff7d8698edff
English	i see.	bf39d91277ed1c47df88ed5771ef8134ceddac75	bf39d91277ed1c47df88ed5771ef8134ceddac75
	billy	7bdf7e5f0ef0f60f0efde5fe0fe5fe0de5e5ee33	051522d0c46404d8ba5b692a10a37b99b8186360
	sterling.	71de0ef90ef0eaaefe0fe5f56dd56df00e07e062	7d2d36108fe26e123cca93300f566c83b3397646

ランダム文字列は, MD5 と同様にどの入力文字列に対しても同じような出力をしていることがわかる. しかし, MD5 と比較して SHA1 は, 1 byte の入力文字列でも推定ができている時と出来ていない時がある. そのため, MD5 より SHA1 の方が学習精度が低い傾向にある. 英文は, MD5 と同様, 短い文字列の時のみ推定が行えていた. 特に, 長い文字列となると, 殆ど同じ文字のみを出力しており, 学習しているとは判断できない結果となった.

4.5 考察

チェックサム及びハッシュ値の学習結果を見ると、学習していない状態のチェックサムの正答率は 0.39%, CRC16 は 0.0015%, CRC32 は 2.32831E-10%, MD5 は 2.93874E-39%, SHA1 は 6.84228E-49%であるため、非常に高い正答率を得ることができた。また、実装に向けた更なるニューラルネットワークの構築が必要であると考えられる。それに伴い、本研究で利用したニューラルネットワークの一種である Encoder・Decoder モデル [2] だが、殆どニューラルネットワークの構築を変更せずに学習を行ったため、改善の余地があると考えられる。近年の機械学習における研究は著しい成長を遂げていることもあり、自然言語処理系ニューラルネットワークは様々なモデルが発表されている。そのため、他のニューラルネットワークを使って学習を行うと、また違う結果が得られる可能性が考えられる。今回、学習済ニューラルネットワークを用いて、入力文字列が可変長である場合で推定を行なったが、固定長で推定を行うとどれほどの正答率が得られるのか検証する必要がある。

第5章 結論

本研究では、ニューラルネットワークを用いたランダム文字列と規則性のある文字列からチェックサム及びハッシュ値の推定を行った。特に、ニューラルネットワークに Encoder・Decoder モデルを用いた。その結果、チェックサムはランダム文字列に対して 20%, 英文に対して 50.8%, CRC16 はランダム文字列に対して 9.01%, 英文に対して 9.08%, CRC32 はランダム文字列に対して 11.9%, 英文に対して 4.17%, MD 5 はランダム文字列に対して 12.07%, 英文に対して 2.53%, SHA 1 はランダム文字列に対して 4.95%, 英文に対して 10.75%の正答率が得られた。この結果から、変異ベース手法でのファジングの対象データでチェックサム及びハッシュ値が採用されていた場合、従来の入力データ通過率よりはるかに高い精度で行うことが可能である。しかし、入力文字列が長い場合推定することが出来ないことが多かったため、より実装に向けた取り組みが必要である。

この研究を通じて、ニューラルネットワークの出力が何通りもある課題に対して、自然言語処理系ニューラルネットワークが有効であることがわかった。一般的なニューラルネットワークで CRC や MD5 を推定しようとしても、出力層が非常に多くなってしまう。それにより、計算コストが非常に高く、時間がかかってしまうため、処置能力が高い環境で行う必要性がある。

今後の課題は、さらなる精度向上に伴い、他の種類のハッシュ値の推定、及び実装評価である。今回は 4 種類のハッシュ値の実験を行ったが、どれも実装するには厳しい精度となっているため、より汎用性を高めることが重要になると考えられる。また、可変長データでの実験しか行っていないため、固定長で推定を行うとどのような結果をもたらすのかなど、多くの検証が必要になる。

謝辞

本研究に際し、多くの方々から御指導、御支援を賜りました。ここに感謝の意を表します。機械学習の研究に携わる機会を与えていただき、研究に関する多くの御指導、御支援を賜りました石浦菜岐佐教授に心より感謝いたします。本研究に関するファジニングに関する御助言等、本研究を進めるにあたり、様々な面で御支援をいただきました難波学之氏に感謝いたします。最後に、研究のみならず生活面においても多くの御支援をいただいた関西学院大学理工学部石浦研究室の皆様へ深く感謝いたします。

参考文献

- [1] 難波 学之: “変異ベースファジリングのためのチェックサムの機械学習,” 関西学院大学理工学部情報科学科卒業論文 (Mar. 2019).
- [2] Ilya Sutskever, Oriol Vinyals, and Quoc V Le: “Sequence to sequence learning with neural networks,” in *Proc. nueral Information Processing systems*, pp. 3104–3112 (Sept. 2014).
- [3] Andrew Tanenbaum, David Weatherroll 著, 水野忠則 訳: “コンピュータネットワーク,” 日経 BP,(Sept. 2013).
- [4] IPUSIRON 著: “暗号技術のすべて,” 翔泳社,(Aug. 2017).
- [5] 林 芳樹 著: “暗号理論入門,” 丸善出版,(Apr. 2012).