本の厚みによる歪みを考慮した光学文字認識

比留川 翔哉[1,a)]　丸山 一貴[1,b)]

概要：近年, 様々な光学文字認識サービスが利用できるが, 歪み補正や, 平滑化などの適切な補正が行われず精度低下の原因になる. 本研究は, 本の厚みによる歪みに近似した画像を生成し, 畳み込みニューラルネットワークを用いて文字認識を行う. 本研究では, 厚みのある本を見開きにした場合に左右に湾曲する歪みを前提とする. そのため, フォントデータから生成した文字画像を OpenCV の remap メソッドを用いて歪ませることで, 学習データを生成した. 提案手法の精度は, 生成した画像のホールドアウト検証では 99.97%となったものの, 撮影した書籍の文字画像では既存手法より劣る結果となった. この結果は, 文字色や背景色などの学習画像を生成する要件と層の最適化が不十分なためだと考えられる. 今後の課題として, 精度の向上を目指していきたい.

キーワード：光学文字認識, 畳み込みニューラルネットワーク, 生成型学習法

1. はじめに

近年 Google Cloud Vision API [1], Adobe Acrobat [2], など, 文字を含む画像から文字コードに変換する光学文字認識 (Optical Character Recognition, 以下, 「OCR」という.) サービスやアプリケーションが存在する. また, Evernote [3], Google Drive [4] などのクラウドにアップロードすることで OCR を使用できるサービスも多く存在する. しかし, 既存の OCR の多くは, 歪んだ画像 (図 1) の文字認識精度を上げるために, 平滑化や歪み補正を行った上で文字認識をする.

平滑化とは, 画像内の精度低下につながるノイズを除去する手法である. 主にガウシアンぼかしや K-最近傍法を使用する. 平滑化を適切に行わなければ, 文字形状の特徴量欠落や, ノイズ増加など, 精度の低下の原因になる. 歪み補正とは, 学習データや事前に取得した特徴量を取得した画像に近似させることである. 歪み補正が適切に行わなければ, 新たな歪みの原因や, 適切ではない文字抽出の原因, 違う文字に近似する原因になり, 精度が低下する. そのため, 画像ごとに手動での平滑化や, 歪み補正のパラメータの変更を行う事で認識率が改善するが, 大量の画像を手動で補正をかけることは難しい. 加えて, 既存の OCR の補正は十分ではないため, 精度の向上は難しい.

本研究では, 誤認識の原因となる補正を行わずに, 本の厚みにより歪んだ文字を認識する OCR を実現することを目的とする. 本研究の OCR は, 生成型学習法を用いて歪んだ文字画像の学習データを作成し, 畳み込みニューラルネットワーク (以下, 「CNN」という.) で学習したものを使用する.

[1] 明星大学情報学部情報学科
Department of Information Science, School of Information Science, Meisei University
[a)] 15j5121@stu.meisei-u.ac.jp
[b)] kazutaka.maruyama@meisei-u.ac.jp

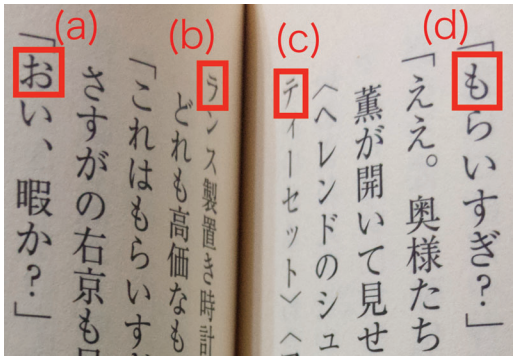


図 1 本の厚みで湾曲した画像 (出典：脚本・輿水泰弘ほか, ノベライズ・碇卯人, ”相棒 season 4 下”, 朝日新聞社, pp.222–223, 2009.)

2. 関連研究

荒木ら [5] は, 書籍を裁断せずにスキャンし, ページ外形や文字行を利用して, 本の厚みによる歪みや輝度・文字ボケの補正を行う手法を提案している. しかしこの手法では, スマートフォンのカメラで撮影した画像や, 白以外のページ色の書籍では正しく補正を行えない. OCR に使用する補正方法を提案しているため異なるため, 本研究とはアプローチが異なる.

志久ら [7] は, 傾斜文字を補正して文字認識を行う手法を提案している. この手法は, 傾斜文字の文字枠を作成し, その文字枠を正方形に補正する. その後に, CNN を用いて文字認識を行っている. しかしこの手法では, 三次元の変形や本の厚みによる湾曲した歪みなどの文字の認識は行えない. 成田ら [6] は, スキャナで取得した歪みのない英数字の文字画像を x,y,z 軸で回転させ, これらを学習し, 看板の文字を認識する手法を提案している. しかし, この手法では, スキャナで歪んでいない文字を取得させてから学習データを作成するため, 認識させる文字種を増やすことが困難である. また, 三次元の回転以外の湾曲した歪みなどの文字認識は精度が低下する. 本研究は, 補正を行わずに文字認識する手法を提案しており, 想定する歪みも異なる.

Ofusa ら [8] は, フォントのグリフを変形させ, 学習データを増やし, CNN の精度を向上させる手法を提案している. この手法は, 補正後の文字認識の学習データの拡張のため, 本研究が対象にしている歪みには有効ではない. また, フォントデータ内のグリフデータを利用して変形させているため, 本研究の画像を用いて変形させる手法とはアプローチが異なる.

石田ら [9] は, 携帯電話のカメラの画像の文字を, 生成型学習法を用いて, 光学的ぼけ, ぶれ, 低解像度に対応する学習データを作成し, 部分空間法と複数フレーム認識, カメラ速度を用いた識別法を使用して文字認識を行う方法を提案している. 本研究は, 学習方法に CNN を使用している点と, 対象にする歪みを湾曲にする点で異なっている.

3. 提案手法

提案手法は, (1) 学習画像生成, (2) 学習, (3) 文字認識の 3 つのフェーズからなる.

3.1 生成型学習法を用いた学習画像生成

生成型学習法とは, 原画像に対して様々な変形や歪み加工などを行い, 学習画像を人工的に生成する手法である. 生成型学習法を用いることにより, 学習データに用いる歪んだ文字画像を撮影せずに学習することが可能である. 本研究では原画像をフォントデータから作成し, 本の厚みによる歪みに近似した変形を行い, 学習画像を作成する. 作成する画像サイズは, 128 × 128 ピクセル, 使用したフォントとサイズは, ヒラギノ角ゴ ProN W3 の 120pt である. 使用する文字種を表 1 に示す. 変形方法は OpenCV_contrib(ver. 3.4.3.18) の remap メソッドを使用する. remap メソッドは, 横方向の変形後の座標と, 縦方向の変形後の座標を格納した 2 つの配列から画像を変形させるメソッドである. remap メソッドは以下の式を使用する. dst は出力画像, src は変換元の画像の座標である.

$$dst(x, y) = src(map_x(x, y), map_y(x, y))$$

remap メソッドの map_x と map_y は, 変形方向によって使用する式を変更した. 使用した式は以下の通りである.

$$map_x(x,y) = \frac{x^{1+\gamma_x}}{h^{\gamma_x}} \tag{1}$$

$$map_x(x,y) = |\frac{(h-x)^{\gamma_x+1}}{h^{\gamma_x}} - h + 1| \tag{2}$$

$$map_x(x,y) = x \tag{3}$$

$$map_y(x,y) = y - \frac{x^2}{w \cdot (n-\gamma_y)} \tag{4}$$

$$map_y(x,y) = y - \frac{(w-x)^2}{w \cdot (n-\gamma_y)} \tag{5}$$

$$map_y(x,y) = y \tag{6}$$

w は画像の横幅, h は画像の縦幅, γ_x と γ_y は歪み定数である. これらの数式は, remap を用いて生成した画像を目視で確認し, 本の歪みに近似しているものを使用した.

作成した学習画像を図 2 に示す. 図 2 は, 上段から原画像 (1 枚), 左下・右下に湾曲させた画像 (25 枚ずつ), 左・右に寄せた画像 (25 枚ずつ), 左側のみ・右側のみ下に湾曲させた画像 (25 枚ずつ), 合計 151 枚である. 学習画像は, 作成した画像を, 背景色を白色, サイズを 128×128 ピクセルに正規化したものである. 使用した map_x 及び map_y の式と想定した歪みを表 2 に示す.

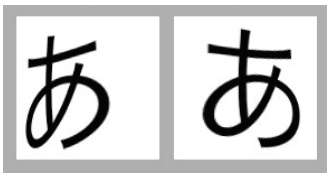
表 1　使用文字種と文字数

文字種		文字数
ひらがな	清音	46 文字*2 種類
カタカナ	濁音・半濁音	25 文字*2 種類
	拗音 (ゃゅょ)	3 文字*2 種類
	促音 (っ)	1 文字*2 種類
	合計	150 文字
アルファベット	大文字	26 文字
	小文字	26 文字
常用漢字 [10]		2136 文字
アラビア数字		10 文字
記号 . ・ - & % 「 」( )		10 文字
合計		2358 文字

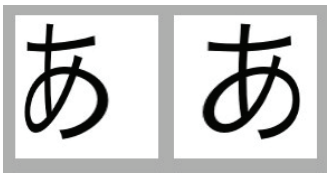
3.2　CNN を用いた学習

本研究は, Keras(Ver. 2.2.4) と TensorFlow(Ver. 1.11.0) を用いて生成した画像を学習する. 本研究で使用する学習画像とテスト画像は, 原画像を除く, 生成型学習法で作成した歪んだ画像 150 枚の内, ランダムに選択した 30 枚をテスト画像, 残りの 120 枚と原画像 1 枚を学習画像とする. 使用した層は VGG16(表 3)[11], 重みの初期値は ImageNet, 使用した活性化関数は, dense_3 は softmax, それ以外は relu, とする. 学習する層は, flatten_1 以下の階層を学習したデータを学習データ 1, block5_conv1 以

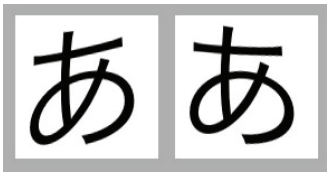
(a) 原画像



(b) 左下・右下に湾曲した歪み



(c) 左・右に寄せた歪み



(d) 左・右側のみ下に湾曲した歪み

図 2　生成する文字画像

表 2　使用した式と種類

歪みの種類	式番号	想定する歪み
右下に湾曲した歪み	(1),(4)	図 1(a),(b)
左下に湾曲した歪み	(2),(5)	図 1(c),(d)
右に寄せた歪み	(1),(6)	図 1(a),(b)
左に寄せた歪み	(2),(6)	図 1(c),(d)
右側のみ湾曲した歪み	(3),(4)	図 1(a),(b)
左側のみ湾曲した歪み	(3),(5)	図 1(c),(d)

下の階層を学習したデータを学習データ 2 とする. epoch 数は, 学習データ 1 を 24000 回, 学習データ 2 は 36000 回とした. epoch 数の選定は, 50000 回のうち最も認識率のよいデータとした. 学習データ 1 と学習データ 2 の精度と誤差関数の値のグラフを図 3, 図 4 に示す. 図 3, 図 4 から学習が正常に行えることが確認できたが, 学習データ 1 の精度は, 80%前後から増加しなかった.

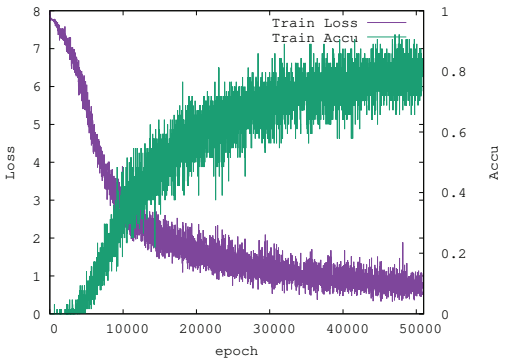


図 3 学習データ 1 の精度と誤差関数の値のグラフ

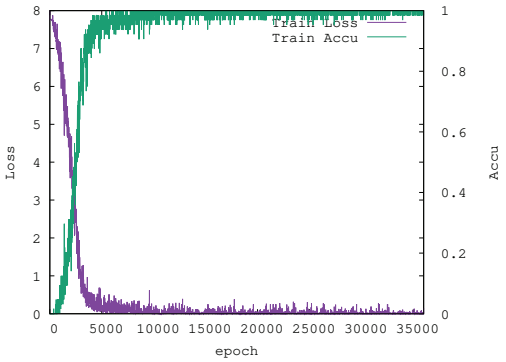


図 4 学習データ 2 の精度と誤差関数の値のグラフ

3.3 文字認識

本手法の文字認識は, 第 3.2 節で作成した学習データ 1, 学習データ 2 の分類器を用いて, 生成型学習法で生成したテスト画像と書籍から撮影した文字画像を認識することで精度を算出する. 書籍の文字を撮影した画像は, 文字の縦と横の大きい方を 1 辺とする正方形に切り抜いた後, 128×128 ピクセルに正規化した上で文字認識を行った. 精度算出に使用した画像は, 第 3.2 節で生成したテスト画像 (生成時に使用した文字全種 70740 枚) と, 小説を撮影した画像 (ひらがな 50 種類 122 枚, カタカナ 30 種類 43 枚, 漢字 61 種類 76 枚) である. テスト画像と, 小説の文字画像は, 無補正で使用する. テストに使用した小説の文字画像の一部を図 5 に示す.

表 3 使用した CNN の層

層の種類	出力のシェープ
input_1 (InputLayer)	(128, 128, 3)
block1_conv1 (Conv2D)	(128, 128, 64)
block1_conv2 (Conv2D)	(128, 128, 64)
block1_pool (MaxPooling2D)	(64, 64, 64)
block2_conv1 (Conv2D)	(64, 64, 128)
block2_conv2 (Conv2D)	(64, 64, 128)
block2_pool (MaxPooling2D)	(32, 32, 128)
block3_conv1 (Conv2D)	(32, 32, 256)
block3_conv2 (Conv2D)	(32, 32, 256)
block3_conv3 (Conv2D)	(32, 32, 256)
block3_pool (MaxPooling2D)	(16, 16, 256)
block4_conv1 (Conv2D)	(16, 16, 512)
block4_conv2 (Conv2D)	(16, 16, 512)
block4_conv3 (Conv2D)	(16, 16, 512)
block4_pool (MaxPooling2D)	(8, 8, 512)
block5_conv1 (Conv2D)	(8, 8, 512)
block5_conv2 (Conv2D)	(8, 8, 512)
block5_conv3 (Conv2D)	(8, 8, 512)
block5_pool (MaxPooling2D)	(4, 4, 512)
flatten_1 (Flatten)	(8192)
dense_1 (Dense)	(4096)
dropout_1 (Dropout)	(4096)
dense_2 (Dense)	(4096)
dropout_2 (Dropout)	(4096)
dense_3 (Dense)	(2358)

4. 実験

実験に使用したコンピュータの構成を表 4 に示す. 作成した文字認識システムと, 既存 OCR である Tesseract の文字認識精度を表 5 に示す. また, 文字認識にかかる実行時間を表 6 に示す. 表 5 の提案手法の括弧内の数字は, 上位 3 位までの累積認識率 (第 3 位累積認識率) を示す. 表 6 の実行時間は, 学習時間を除いたテスト画像 70740 枚と小説

図 5　小説の文字画像

の文字画像 241 枚の認識にかかった時間である.

表 4　実験で使用したコンピュータの構成

パーツ	名前
OS	Ubuntu 16.04
CPU	Intel Core i7-7700 @ 3.60GHz
メモリ	DDR4-2666MHz 32GB
GPU	GeForce GTX 980 Ti 6GB

表 5　認識結果

OCR の種類	テスト画像	小説の文字画像
学習データ 1	98.94% (99.98%)	33.61% (49.79%)
学習データ 2	99.97% (100%)	49.38% (61.41%)
Tesseract	77.54%	83.40%

表 6　文字認識にかかる時間

OCR の種類	テスト画像	小説の文字画像
学習データ 1	13 分 12 秒	5 秒
学習データ 2	12 分 20 秒	5 秒
Tesseract	9 時間 26 分 24 秒	1 分 50 秒

5. 考察

表 5, 表 6 の学習データ 1,2 から総結合層 (dense 層)・flatten 層・dropout 層のみ学習させた場合と, block5 の 4 層を追加した場合では, 16%の精度の差と 1 分近い実行時間の差があった. また, 全層学習[*1]させた場合, epoch 数 500000 まで学習させたが, 正しく学習できなかった. 原因は, 層の数や層の入力数と出力数が文字認識のモデルと相性が悪いと思われる. 改善策は, 層の数と層の入力数と出力数の調整を行うことで, テスト画像の認識精度が 100%に近づくと考えられる.

表 5 のテスト画像と小説の文字画像の精度の差から, 書籍の文字画像を認識する際に必要な要件不足と, CNN の層が最適化されていないと考えられる. 精度向上に必要な要件は,

- 背景と文字の輝度対比が大きい画像を生成すること
- 本の見開き中央のピントボケの画像を生成すること
- 背景色・文字色を増やすこと
- 様々なフォントで画像を生成すること
- 縦書きと横書きの文字位置を想定して画像を生成すること

が考えられる. 輝度対比と背景色・文字色は複数の背景色と文字色の組み合わせが考えられる. 考慮すべき背景色は, 白色, 黒色, 赤色, 緑色, 青色, 文字色は黒色と白色が挙げられる. 本の見開き中央のピントボケは石田ら [9] の手法を用いることでピントボケの画像が生成できる. フォントは「ゆ」や「そ」などフォントによって文字形状が変わる場合があるため考慮すべきである. 丸括弧や鍵括弧, 拗音など縦書きと横書きで表示位置が違う文字は, 縦書きと横書きの画像を生成する必要がある. これらの要件の影響を調査し, 学習画像の追加等を検討していく.

現在は, VGG16 の層を使用しているが, これらは物体識別で使用された層である. 文字認識は, 必要な色情報が最低 2 色な点と, 形状把握に必要な解像度が小さい点から, 必要な層の数が少ないと思われる. そのため, 層の数の調整や Dropout 層や flatten 層の追加などの最適化が可能である. その最適化により, 学習速度の向上や, 精度の向上, 識別文字数の増加などが期待できる.

表 6 の文字認識にかかる時間の差は, 提案手法の文字認識方法と Tesseract の文字認識方法の違いから発生するものと考えられる. 提案手法は, 入力画像の全体を判別器に使用して文字認識を行っているが, Tesseract においては, 入力画像から文字形状を抽出した上で文字認識を行うため, 提案

[*1] 全ての層を学習させる手法

手法より処理に時間がかかる.

6. まとめ

本論文では, 本の厚みで湾曲した文字画像を無補正で認識するため, 生成型学習法を用いて学習画像を生成し, CNN を用いて文字認識する手法を提案した. 本手法は, フォントデータから作成した画像から, 本の厚みで湾曲した文字を近似した学習画像を作成し, 学習と文字認識を行う. 提案手法は既存の OCR である Tesseract に比べて, 生成型学習法で生成したテストデータには有効性が確認できたが, 小説を撮影した文字では有効性が確認できなかった. 今後の課題として, 学習に用いた CNN の層の最適化や, 学習画像の生成の際に使用する要件の追加を行い精度向上を目指す. また, 学習画像の生成に使用するフォント, 文字色, 背景色などを変更した画像の追加による精度の影響を調査して, 検討する.

参考文献

[1] Google, CLOUD VISION API, <https://cloud.google.com/vision/> (参照 2018/08/26).

[2] Adobe, Adobe Document Cloud, <https://acrobat.adobe.com/jp/ja/acrobat.html> (参照 2018/08/26).

[3] Evernote, Evernote, <https://evernote.com/intl/jp>(参照 2018/08/26).

[4] Google, Google ドライブ, <https://www.google.com/intl/ja_ALL/drive/> (参照 2018/08/26).

[5] 荒木禎史, 関海克, 小島啓嗣, ”製本原稿スキャン画像の歪み補正技術”, Ricoh Technical Report, No.29, pp.31–40, 2003.

[6] 成田了, 大山航, 若林哲史, 木村文隆. ”3 次元回転不変カメラベース文字認識”, 電気学会論文誌 C (電子・情報・システム部門誌), 133 巻, 4 号, pp.876–882, 2013.

[7] 志久修, 手島裕詞, 内田誠一, ”傾斜文字認識のための正規化方法”, 電子情報通信学会論文誌 D, Vol.J100-D, No.10, pp.902–906, 2017.

[8] Kenichiro Ofusa, Tomo Miyazaki, Yoshihiro Sugaya, Shinichiro Omachi, ”Glyph-Based Data Augmentation for Accurate Kanji Character Recognition”, 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, pp.597–602, 2017.

[9] 石田皓之, 高橋友和, 井手一郎, 目加田慶人, 村瀬洋, ”携帯カメラ入力型文字認識におけるぼけやぶれに対処するための生成型学習法”, 電子情報通信学会論文誌 D, Vol.J89D, No.9, pp.2055–2064, 2006.

[10] 文化庁, 常用漢字表 (平成 22 年内閣告示第 2 号) <http://www.bunka.go.jp/kokugo_nihongo/sisaku/joho/joho/kijun/naikaku/kanji/index.html>(参照 2018/08/22).

[11] Visual Geometry Group, Very Deep Convolutional Networks for Large-Scale Visual Recognition <http://www.robots.ox.ac.uk/~vgg/research/very_deep/>(参照 2018/11/28).