機械学習を用いた雨滴感知によるワイパーの自動起動の改善

2020SS013 堀恭瑳
指導教員：小市俊悟

1　はじめに

日本には小雨が多い時期があるが，車のフロントガラスが少量の雨で濡れているとき，逐一ワイパーを手で作動させることは煩わしいと感じる．しかし，一方でワイパーを一定間隔で作動させると，小雨ではフロントガラスがそれほど濡れていないときもワイパーが作動してしまい，ワイパーそのものを傷つけてしまうこともある．したがって，結局，手で雨に応じてワイパーを停止させることが必要である．このようなことから自動でワイパーを起動および停止させる機能を備えた自動車も増えてきたが，そのワイパーの起動・停止の判定には満足できないこともしばしばある．雨量を感知するセンサーには様々なものが存在し，例えば，雨滴による車のフロントガラスでの反射光の屈折率の変化を感知するようなセンサーがある．本研究では，そのようなセンサーによる判定を補助することを目的に，画像について近年良い成果を上げている機械学習を用いて，カメラで雨滴を感知する方法の確立を目指す．

2　研究方法

機械学習の中でも，画像に関する課題に対して様々な成果を上げているニューラルネットワークを用いて，フロントガラスが濡れているか濡れていないかを判断するプログラムを作成する．本研究では，具体的なニューラルネットワークとして scikit-learn の MLPClassifier を用いた．

機械学習のためのデータを収集するために，自動車のフロントガラスを想定した透明なアクリル板を用意し，それを濡らした画像と濡らしていない画像をそれぞれ 100 枚ずつ用意した．濡らした画像と濡らしていない画像を図 1 と図 2 にそれぞれ 1 枚ずつ示す．

図 1　濡れている画像

図 2　濡れていない画像

これらの画像をグレースケールに変換したものも用意して，機械学習を行う [1].

最終的なシステムの流れとして，フロントガラスをカメラで検知し，雨等で濡れたことを感知する．感知した場合，ワイパーを起動する信号を発生させることで雨等をふき取る．濡れていないのであれば，ワイパーを一時停止する信号を発生させる．

3　実験結果

学習データに対する正解率は表 1 のような結果になった．実験はカラー画像のままで行った場合とグレースケール画像に変換して行った場合がある．

表 1　カラー画像とグレースケール画像に対する正解率

試行	1 回目	2 回目	3 回目
カラー画像	80.69%	69.31%	70.30%
グレースケール画像	100%	100%	100%
試行	4 回目	5 回目	6 回目
カラー画像	100%	50.00%	58.91%
グレースケール画像	100%	90.05%	100%

表 1 の結果から，グレースケール変換をした方が精度が向上することがわかる．カラー画像の場合も高い正解率を示す時もあるが，グレースケール画像の方が安定的に高い精度を示す．実際，人間の目にもグレースケール変換をした方がカラー画像より雨をはっきりと視認することができ，その結果精度が向上したと考えられる．

ここまでは学習データに対する精度を検証した．しかしそれだけでは正確な精度とはいえない．新たに検証用のデータを作成し，それに対する精度を評価することで，実際に近い環境にすることができる．ただし，検証データも学習データと同じような環境で撮影したものであり，人間の目には似かよったものである．

今回は 12 枚からなる検証用データを作成し，それに対する正解率を求めたところ，表 2 のようになった．

表 2　検証用データに対する正解率

試行	1 回目	2 回目	3 回目
カラー画像	40.00%	85.00%	80.00%
グレースケール画像	50.00%	62.50%	50.00%
試行	4 回目	5 回目	6 回目
カラー画像	50.00%	50.00%	85.00%
グレースケール画像	37.50%	37.50%	75.00%

表 2 の結果から，精度が低い時が多く，安定した結果を出し続けられないことがわかる．また，グレースケール画像の場合の 4 回目や 5 回目のように精度が低くなることを防ぐために，学習させる画像を増やす必要もある．そこで，学習データを増やし，利用する機械学習もより高い精度を出力することができる深層学習に変更して検証することにした．

4　変更点

深層学習には PyTorch を用いた．また，より判別精度を上げるために濡れている画像を 355 枚と濡れていない画像を 405 枚用意した．濡れていない画像の枚数は学習を行った後にさらに精度を高めるために追加したので，濡れている画像に対して多くなっている．これらの画像を訓練用データ，検証用データ，テスト用データの 3 つに分類する．テスト用データは実際に学習させていないデータにして学習結果の精度を計算したいため，新しく濡れている画像と濡れていない画像を 15 枚ずつ撮影したものを入れている．

5　手順

以下の Step1 から Step5 を用いて学習する [2].

Step1：訓練用データからランダムに画像のサイズ，左右反転，回転をするような transforms クラスを作成する．

Step2：Dataset を作成する．

Step3：DataLoader を作成する．

Step4：nn.Module を継承して，畳み込み層，最大値プーリング層，dropout を 4 回繰り返した後，全結合層を 1 つおいて分類するような深層ネットワークのモデルを作成する．

Step5：Step1〜4 まで利用し学習を実行する関数を作成し，エポック数を指定して学習させる．

これらを行った後，精度の良い学習ができているかを確認するために，正解率の変化と損失の変化をグラフに表す．また，学習結果を再利用するために，学習済みのモデルを保存する．実際に使えるかを確認するために，検証データに対する正解率を算出する．

6　深層学習の実験結果と考察

エポック数を 100 回とし，3 回学習をした．3 回とも似ているグラフになったので，3 回のうちの 1 回の正解率を図 3 に損失関数値を図 4 に示す．

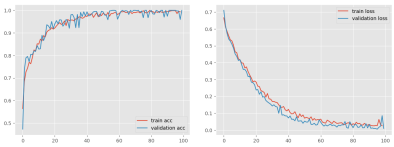

図 3　正解率　　図 4　損失関数値

検証データに対する正解率は 77.42%，80.65%，80.65% となった．

エポック数を 300 回とし，3 回学習をした．3 回とも似ているグラフになったので，3 回のうちの 1 回の正解率を図 5 に損失関数値を図 6 に示す．

検証データに対する正解率は 80.65%，83.87%，87.10% となった．

学習回数が 300 回の方が精度が良かったため，その深層ネットワークを用いて実際にカメラをを起動しリアルタイムで判定させた．その過程で得られた画像が図 7 から図 9 である．

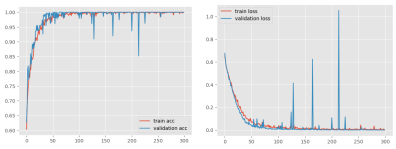

図 5　正解率　　図 6　損失関数値

図 7　濡れていると誤判定された画像　　図 8　濡れていないと正解した画像　　図 9　濡れていると正解した画像

濡れている場合はかなりの精度で正しく判定出来ていることがわかる．しかし，濡れていない場合でもときどき濡れているという判定になってしまう．これが原因で精度が少し悪くなっていると考えられる．これを解決するためには，質の良い濡れていない時の学習画像を増やすことが挙げられる．

7　おわりに

本研究では，ワイパーの起動・停止のセンサーを補助することを目的として PyTorch を用いた深層学習を行い，雨滴を感知する方法の確立することに取り組んできた．実行結果から PyTorch の深層学習を用いた方がより精度の良い判定を行うことができた．ただし，依然として深層学習を行ったモデルを用いて実際にカメラを起動させ判定させたとき，濡れていないにも関わらず，濡れていると間違った判断をする時がある．

本研究では，天気によって背景が変わることや，夜のときに判断しにくくなることが予想される．しかし，これらの特徴を含めた学習画像を増やすことが出来れば，さらに精度が高くなり，対応できる状況も増えると考える．それが出来れば，実際に車の機能として利用することができると考えている．

参考文献

[1] nkmk.me：『Python, OpenCV, NumPy で画像を二値化（しきい値処理）』https://note.nkmk.me/python-numpy-opencv-image-binarization/ (2023/09/11)

[2] ｔｋ：『[Pytorch] じゃんけん画像を分類してみた』https://zenn.dev/takeguchi/articles/672ff3b34753a7 (2023/12/19)