Reconhecimento de Placas de Carro

Lucas H. M. Silva
Departamento de Computação
Universidade Federal de Ouro Preto - UFOP
Ouro preto, MG, Brasil
lucashenrique580@gmail.com

Samuel Q. S. Rocha
Departamento de Computação
Universidade Federal de Ouro Preto - UFOP
Ouro preto, MG, Brasil
samuelqsrocha@gmail.com

***Resumo*—O trabalho propõe a avaliação de diferentes técnicas de binarização aliadas a técnicas de *Machine Learning* para realizar a segmentação e reconhecimento de caracteres de placas de carro para uma mesma base de placas que possuí um volume de dados reduzido.**

***Keywords*-Manipulação de Imagem, *Machine Learning*, LPR.**

I. Introdução

O Reconhecimento de Placas de Carro (ou LPR do inglês License Plate Recognition) consiste em um sistema que envolve a captura de uma imagem de um carro e o processamento dessa imagem para obtenção do conteúdo da placa do carro. Com isso, pode-se perceber que é uma aplicação que envolve diversas áreas, desde engenharia de sensores até desenvolvimentos de aplicações para apresentação dos dados processados. Esse trabalho irá se focar nas etapas de detecção e reconhecimento da placa numa imagem. Assumindo que ela já foi capturada e pré processada de modo que as imagens sempre terão placa visível.

O LPR é fundamental em sistemas de apoio a autoridades de trânsito de uma grande cidade, por exemplo, onde muitas vezes é importante reconhecer a placa de um veículo para garantir uma determinada fiscalização ou aplicação de uma penalidade ao dono de um veículo que por sua vez está associado ao registro da placa. Outro cenário de aplicação de LPR é em estacionamentos, no qual é importante manter um registro dos carros que entram e saem do local. Além disso, automatizar o reconhecimento das placas dos veículos agregaria valor ao estabelecimento, pois poderia reduzir a mão de obra e agilizar o processo de entrada e liberação de veículos.

Do ponto de vista do processamento de imagens, a etapa de detecção e reconhecimento da placa já apresenta vários desafios: Como localizar a placa dentro da imagem sem ambiguidade com outras áreas do carro? Como reconhecer os caracteres que compõe o registro do veículo? Como estavam as condições do tempo quando a imagem foi capturada (chuvoso, nublado, ensolarado etc)? Como fazer todo esse processo de maneira performática, para que se obtenha resultados em tempo real?

A. Trabalhos relacionados

Para atacar esses problemas, vários trabalhos foram desenvolvidos aplicando diversas técnicas. A maioria deles usa técnicas de manipulação de imagens para poder encontrar a placa e posteriormente usam alguma técnica de *Machine Learning* para reconhecer os caracteres [1] [2] [3][4]. Outros trabalhos se apoiam exclusivamente em técnicas de Aprendizado de Máquina [5] enquanto outros se valem de processamentos estatísticos na imagem para resolver o pipeline de processamento do LPR [6].

Em [1] o foco do trabalho é apresentar uma nova abordagem ao Reconhecimento de Placas de Carro. O objetivo é que a abordagem possua a menor quantidade possível de restrições quanto às características que as imagens de entrada devem ter. Imagens onde a as placas estão danificadas, sujas ou fora do ângulo de visão são consideradas como entradas válidas e tem bom resultado na maioria dos casos. O processo de reconhecimento é dividido em duas etapas: localização da placa e identificação dos caracteres da placa. Para a primeira etapa, a imagem de entrada tem suas bordas detectadas por técnicas já conhecidas. Depois o espaço cores é transformado no espaço HSI. Por fim uma técnica fuzzy é usada para determinar as partes da imagem que pertencem a uma possível placa dado as características conhecidas de uma placa (formato, cor etc.). Para a segunda etapa, os possíveis caracteres de dentro da área apontada pelo primeiro passo têm suas características topológicas extraídas e comparadas com as mesmas características de templates de caracteres já conhecidos. Dentre todos os templates compatíveis, o que melhor representa o caractere da imagem é determinado através de uma rede neural. Depois de testes feitos com imagens de cenas complexas com objetos parecidos com placas, diferentes ambientes e condições de iluminação e placas danificadas, a acurácia geral da abordagem foi de 93.7%. De acordo com os próprios autores, o algoritmo de identificação de caracteres teve dificuldades em distinguir “1” de “7”. Além disso, quando existem múltiplos candidatos à placas na mesma imagem o tempo de processamento pode ser proibitivo para aplicações em tempo real. Por outro lado, a técnica apresentada traz diversas contribuições como descartar bordas irrelevantes num estágio inicial e os templates considerados para comparação são limitados, e com isso têm-se uma boa performance.

Em [2] é proposto uma abordagem que usa bordas verticais para extrair e identificar Placas de Carros em imagens escala de cinza. Apesar de se focar em placas coreanas, as técnicas empregas podem ser facilmente usadas em outros domínios de placas de carro. Na etapa de extração da placa, a imagem passa por filtros que extraem suas bordas e uma nova imagem

é formada somente pelas bordas verticais da imagem filtrada. Depois disso as várias bordas verticais são checadas contra um conjunto de características pré definidas que devem conter para serem consideradas como parte de uma placa. A última etapa é checar cada par de bordas verticais restantes que podem delimitar a área de uma placa quanto a presença de características como a razão altura e largura. Depois disso a região dada como sendo da placa é segmentada em regiões de caracteres de acordo com o formato conhecido das placas. Por fim, na etapa de identificação de caracteres, um algoritmo de checagem de template é usado para comparar os caracteres extraídos com os possíveis caracteres de uma placa válida. A taxa de sucesso do algoritmo proposto é de 94.4% dentro dos testes realizados pelos autores com centenas de imagens em vários pontos de vista e condições. Segundo os próprios autores, a maior desvantagem da abordagem é quando as imagens não têm as bordas bem definidas, o que pode levar a uma segmentação completamente inválida. Quando objetos na frente da placa, as bordas da mesma podem ser removidas como ruído.

Em [6] é apresentada uma abordagem para reconhecimento de caracteres de placas europeias no qual é dividido apenas em treino e validação. O foco do paper é apresentar uma técnica que necessite de poucos exemplos destinados para treino e obtenha um bom resultado. O paper foi desenvolvido na Itália e os autores explicam que na Europa há um grande conjunto de placas diferentes devido ao número de países. Com isso há uma dificuldade em se obter uma base de dados grande para todos os países que podem transitar, por exemplo, na Itália. Surge então esta necessidade de um sistema que necessite de poucos exemplos para treino. A técnica apresentada foca em criar padrões para cada caractere, utilizando distribuições estatísticas e fazendo correções de transformações determinísticas que, de acordo com os autores, são responsáveis por grande parte da variação presente entre as imagens de entrada. O ponto chave é que com poucos, mas bons exemplos os padrões podem ser criados eficientemente. Para realizar uma classificação em seguida, os caracteres são analisados em todos os padrões e um resíduo é calculado. O padrão que obtiver o menor resíduo é selecionado. Os testes foram feitos de duas formas. Na primeira uma base grande foi utilizada para fazer uma comparação de resultados com outras técnicas. Foram utilizadas 7000 placas de diferentes países e obteve um resultado de 98,1%. No segundo teste foram utilizadas apenas 37 exemplos, selecionados manualmente, ou seja, foram selecionados bons exemplos e obteve um resultado de 97,3%.

Em [5] é apresentada uma abordagem para reconhecimento de placas coreanas no qual é dividida em 3 módulos. O primeiro módulo é o de Detecção de Carro, no qual dado uma imagem o sistema identifica uma região onde o carro provavelmente se encontra caso exista um carro na imagem. Este módulo é utilizado para fazer uma filtragem inicial, no qual apenas imagens contendo carros passarão para a próxima etapa, economizando tempo de processamento. Neste módulo é utilizado para identificar os carros a taxa de mudança de cor no sensor vertical. O sensor vertical é uma região da imagem que indica o limite(borda) do carro passando em frente ao sensor que capta a imagem. O segundo módulo é o de Segmentação da Placa, no qual recebe como entrada as imagens selecionadas pelo primeiro módulo e identifica nesta imagem a região que contém a placa do carro. Nesta etapa são utilizadas duas redes neurais(TDNNs). Uma das redes neurais é responsável pela análise vertical e a outra pela análise horizontal. Em seguida é utilizado um pós processador que combina a análise feita pelas duas redes neurais e elimina ruídos das análises. Com essa combinação, baseado no tamanho, formato e outras características das placas de carros, o módulo identifica a placa contida na imagem de entrada. O terceiro módulo é o de Reconhecimento da Placa, no qual recebe como entrada uma imagem de placa selecionada pelo segundo módulo e faz a identificação dos caracteres contidos na placa. Uma placa coreana é dividida em duas linhas, sendo a primeira formada por dois caracteres e um ou dois números, e a segunda um caractere e quatro números. Para fazer a identificação de caracteres é utilizado quatro SVMs, uma para os caracteres da linha de cima, uma para os números da linha de cima, uma para o caractere da linha de baixo e uma para os números da linha de baixo. O sistema foi treinado e validado utilizando 1000 video clipes, sendo 600 para treino e 400 para validação. A taxa de acertos final do sistema foi de 94,7%, no qual o primeiro módulo obteve 100% de acertos, o segundo módulo 97,5% e o terceiro 97,2%. Mais da metade do tempo de processamento foi gasto no segundo módulo, tendo o terceiro módulo como o segundo mais custoso em termos de tempo de processamento e o primeiro módulo tendo o tempo desprezível em relação aos outros dois módulos.

Em [3] é apresentada uma abordagem para reconhecimento de placas chinesas no qual é dividida em 5 fases. A primeira fase é responsável por um pré processamento no qual a imagem é convertida para escala de cinza e em seguida usado um histograma equalizado para melhorar o contraste da imagem. A segunda fase é realizada uma binarização do mapa de borda utilizando o método de Otsu's global thresholding, e é importante pois as próximas fases dependem desse processamento. A terceira fase é responsável pela detecção da placa na imagem e é dividida em alguns passos. O primeiro passo é encontrar na imagem regiões que possivelmente contém a placa baseado em características como, por exemplo, o formato. O segundo passo é a detecção de bordas no qual as partes candidatas são transformadas em imagens apenas com as bordas em destaque. No terceiro passo são calculados vetores que indicam a distância entre bordas em cada linha das partes candidatas. No quarto passo são encontrados os limites esquerdo e direito da placa utilizando os vetores calculados no passo anterior. A escolha de qual parte candidata é a mais provável de ser a placa é feita agora, baseado na maior variação dos vetores. Em seguida inicia-se a quarta fase responsável por identificar o ângulo para rotacionar a placa para que ela fique completamente na horizontal, pois as imagens de entrada não necessariamente estão alinhadas. A rotação também é feita nessa fase. A quinta e última fase é responsável por identificar

os caracteres, no qual é realizado fazendo cortes horizontais e verticais utilizando técnicas que se baseiam em conhecimentos prévios das estruturas das placas. Os experimentos foram realizados utilizando 380 imagens das quais 15 não tiveram as placas encontradas corretamente, obtendo um acerto de 96,1%. Das 365 imagens que a placa foi reconhecida corretamente, 7 não obtiveram uma segmentação de caracteres correta, obtendo um acerto de 94,2%.

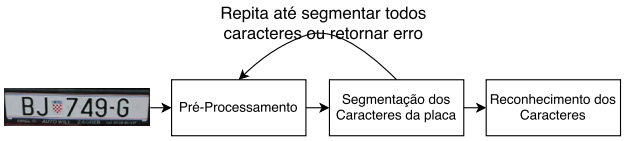

Figura 1. Fluxo básico do processamento de uma imagem de entrada no trabalho desenvolvido

B. Visão geral da técnica

O trabalho proposto apresenta um sistema de LPR que reconhece os caracteres de uma placa baseando-se no conhecimento prévio de características da placa. São utilizadas técnicas de manipulação de imagem para realizar pré processamentos, bem como realizar a segmentação. Uma vez segmentados os caracteres, são usadas técnicas de *Machine Learning* para reconhecimento destes. A contribuição é apresentar uma comparação de desempenho entre várias técnicas, mostrando quais combinações de técnicas obtiveram melhores resultados para uma mesma base de dados. A base de dados utilizada não possui nenhum padrão quanto a quantidade exata de caracteres numa placa ou ordem de letras e números, o que facilita a utilização das técnicas apresentadas neste trabalho em outros domínios de placas de carro.

Além disso, o trabalho se limita às etapas de detecção e reconhecimento dos caracteres da placa, partindo do ponto onde a placa já foi localizada. Dessa forma, as imagens consideradas possuem somente o recorte da placa, sem ruído ou objetos de fundo.

Por tanto, podemos dividir o trabalho em três grandes módulos: pré-processamento da imagem da placa, segmentação dos caracteres e reconhecimento dos caracteres.

II. Fundamentação teórica

Podemos dividir o trabalho em três grandes módulos: pré-processamento (1), segmentação dos caracteres (2) e reconhecimento dos caracteres (2). No módulo 1, recebe-se uma imagem colorida de uma placa visível. A imagem é então convertida em escala de cinza, para eliminar informações irrelevantes, e binarizada, com o objetivo de facilitar identificação de componentes conexos. Tais componentes, idealmente, representam os caracteres da placa. No segundo módulo, os componentes conexos são de fato localizados. Uma vez localizados, precisam ser filtrados para eliminar as regiões que não representam caracteres. Já no ultimo modulo, essas regiões são lidas como imagens individuais e servem de entrada para um modelo de classificação que utiliza técnicas de *Machine Learning*, como, por exemplo, SVM. Tal modelo classifica as imagens e retorna os caracteres identificados. A Figura 1 mostra o fluxo básico da abordagem.

A. Pré Processamento

A imagem capturada, originalmente em RGB, deve ser tratada a fim de que se possa identificar os caracteres com mais facilidade.

O primeiro tratamento é converter a imagem em escala de cinza. Isso é feito, pois dessa forma eliminamos informações de cores que são irrelevantes para nossos objetivos. Em seguida, a imagem sofre uma equalização de histograma para melhorar o contraste.

A última etapa do pré processamento é a binarização da imagem. Com isso, podemos tratar imagens com diversas condições de luminosidade, além de ser necessário para encontrar os componentes conexos. Os métodos utilizados para realizar a binarização são Limiarização Local ou Limiarização de Otsu. Outra etapa é aplicar ou não uma operação morfológica de fecho sobre a imagem binarizada. Para cada imagem são testadas várias combinações dessas operações de modo que a operação final é a operação que permite a detecção de todos os caracteres pelo módulo seguinte.

São considerados esses dois métodos de binarização (Otsu e Local), pois não se pode assumir nada a respeito da qualidade de iluminação e contraste da imagem. Dessa forma, a limiarização de Otsu pode gerar resultados indesejáveis quando a imagem de entrada tem diferenças muito grandes de iluminação.

Bem como as alternativas de binarização, a imagem pode ou não sofrer uma operação de fecho. Isso se faz necessário, pois o limiar encontrado na binarização pode deixar as regiões dos caracteres muito unidas ou ainda unir regiões das bordas com os caracteres. Por outro lado, aplicar o fecho sobre uma imagem que teve uma binarização muito "fina"pode separar partes dos próprios caracteres.

B. Segmentação dos caracteres

Nessa etapa, a metodologia apresentada recebe do módulo anterior uma imagem binarizada e deve segmentar todos os caracteres presentes na placa.

Para obter a imagem binarizada através do módulo de pré processamento, a etapa de segmentação testa exaustivamente cada combinação do Módulo 1. Primeiramente a segmentação é feita na imagem pré-processada com limiarização de Otsu e operação de fecho, caso a quantidade de regiões filtradas seja igual a quantidade de caracteres que o label da placa possui, tais regiões são passadas adiante para o módulo de detecção. Caso a quantidade de caracteres identificados seja diferente, faz-se a segmentação usando a limiarização local e fecho. No caso do segundo método detectar todos os caracteres, a placa segue para a fase de reconhecimento, caso contrário são feitas mais duas tentativas aplicando em sequência a limiarização de Otsu e Local sem operação de fecho. Caso nenhum dos métodos consiga localizar todos os caracteres da placa, essa imagem não passa para etapa de reconhecimento dos caracteres e é retornado um erro.

Na segmentação em si, busca-se os componentes conexos da imagem. Depois de se obter todos esses componentes, é necessário fazer uma filtragem dentre todas as regiões para garantir que apenas regiões que correspondem a caracteres sejam detectadas. Tal filtragem é realizada através das propriedades dos caracteres em relação à área da placa. As propriedades consideradas são:

1) Proporção entre a altura do caractere e a altura da placa completa deve variar entre 45% e 80%;
2) Proporção entre a largura do caractere e a largura da placa completa deve variar entre 4% e 25%;

A Figura 2 mostra um exemplo de saída desse módulo.

Uma complicação que surge a partir desse processo é o fato de que quando as componentes da placa são detectadas, pode ocorrer de que as regiões filtradas como caracteres estejam fora de ordem em relação a posição real dos caracteres na imagem. Isso pode ser resolvido armazenando-se a coordenada inicial de cada região de caractere. Com isso, ao final de todo processo, para se obter os caracteres na ordem correta, basta ordená-los de acordo com as coordenadas armazenadas anteriormente.

Um problema encontrado, devido ao modelo das placas da base utilizada no qual possuiam bandeiras e emblemas juntos aos caracteres, foi que tais símbolos as vezes eram identificados como sendo caracteres, ocasionando uma falha na verificação da quantidade de caracteres encontrados e a quantidade de caracteres do label.

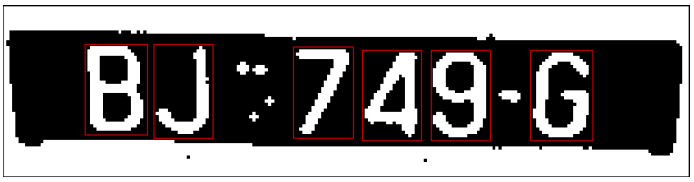

Figura 2. Resultado esperado da segmentação dos caracteres

C. Reconhecimento dos caracteres

Assumindo que a etapa anterior retornou um conjunto de regiões que correspondem a todos os caracteres corretamente, o terceiro e último módulo deve, para cada caractere segmentado, reconhecer qual é a letra ou número correspondente.

A abordagem escolhida para essa etapa faz uso de técnicas de aprendizado de máquina supervisionado afim de treinar um modelo a reconhecer caracteres alfa numéricos a partir de imagens de caracteres.

Para se obter uma base de treino suficientemente boa, as imagens de cada caractere foram degradadas com filtros, transformações afins e ruídos para que possam refletir os diversos estados de iluminação, ângulos da câmera e qualidade de captura no mundo real. Com isso, o treinamento do modelo pode cobrir uma grande gama de situações possíveis de imagens que podem ser reconhecidas com sucesso, uma vez que é impossível predizer com precisão quais serão as condições de iluminação que uma imagem de entrada apresentará.

Durante o treinamento as imagens de cada caractere são carregadas e quatro cópias são cridas. Em cada cópia, aplica-se uma combinação de operações (as mesmas quatro mencionadas na Subseção B). Isso é necessário, pois não se sabe a priori para um determinada placa que está sendo reconhecida qual combinação de operação foi realizada na etapa de segmentação para gerar as regiões dos caracteres.

Dessa forma, a base de dados foi dividida com 90% das imagens para treinamento e 10% para validação. Na Tabela I podemos ver o resultado do reconhecimento dos caracteres da Figura 2

Foram utilizados 7 técnicas para realizar as classificações de caracteres sendo elas: SVM (*Support Vector Machine*) com kernel não linear RBF (*Radial Basis Function*), SVM com kernel linear, K-Neighbors, SGD (*Stochastic Gradient Descent*), MLP (*Multi-layer Perceptron*) com função de erro LBFGS, MLP com função de erro SGD e MLP com função de erro ADAM.

Tabela I
RESULTADO ESPERADO DA CLASSIFICAÇÃO DOS CARACTERES DA PLACA DA FIGURA 2

B	J	7	4	9	G
'B'	'J'	'7'	'4'	'9'	'G'

III. TESTES E RESULTADOS

Para testar o desempenho da abordagem apresentada, desenvolvemos a metodologia na linguagem de programação Python usando as bibliotecas: sci-kit-learn, sci-kit-image-, numpy e matplotlib.

Foi medido quantas imagens foram segmentadas corretamente e qual a combinação de técnicas mais foi usada para tal. Além disso, para cada modelo treinado, foi calculada a proporção entre quantas placas foram reconhecidas por completo (todos caracteres corretos) e o total de placas. Ainda para cada modelo, medimos a taxa de acerto médio por caractere individualmente e a taxa de acerto médio por placa (quantos porcento dos caracteres de uma placa cada modelo acerta em média).

A base de dados possuía 240 imagens de placas, majoritariamente da croácia com diversas condições de iluminação e alinhamento. Ela foi extraída de [7].

Na primeira métrica foi verificado que 90% das placas foram segmentadas corretamente, ou seja, em 90% dos casos as imagens tiveram a quantidade correta de regiões conexas detectadas. A combinação de técnicas que mais funcionou para essa segmentação foi a Limiarização de Otsu em conjunto com a operação morfológica de fecho, que foram usadas em 70% das vezes que os caracteres foram detectados corretamente.

Nas métricas relacionadas aos modelos, foi visto que usando os 7 modelos treinados em conjunto, eles reconheceram corretamente todos os caracteres de 58% das imagens. O modelo que obteve o melhor resultado, tanto no reconhecimento de todos os caracteres de uma placa como no reconhecimento geral de caracteres, foi o SVM com kernel não linear RBF

que obteve 39% de acurácia para reconhecimento completo de uma placa e 86,91% no reconhecimento geral de caracteres. O modelo MLP com função de erro ADAM obteve 34% de acurácia para reconhecimento completo de uma placa e 84,43% no reconhecimento geral de caracteres. O modelo MLP com função de erro LBFGS obteve 30% de acurácia para reconhecimento completo de uma placa e 82,58% no reconhecimento geral de caracteres. O modelo K-Neighbors obteve 30% de acurácia para reconhecimento completo de uma placa e 81,57% no reconhecimento geral de caracteres. O modelo SVM com kernel linear obteve 29% de acurácia para reconhecimento completo de uma placa e 82,56% no reconhecimento geral de caracteres. O modelo MLP com função de erro SGD obteve 28% de acurácia para reconhecimento completo de uma placa e 82,59% no reconhecimento geral de caracteres. Com o pior resultado, tanto no reconhecimento de todos os caracteres de uma placa como no reconhecimento geral de caracteres, foi o SGD que obteve 20% de acurácia para reconhecimento completo de uma placa e 78,17% no reconhecimento geral de caracteres. Todos os resultados podem ser vistos na Figura 3.

O desempenho por caractere do modelo SVM com kernel RBF pode ser visto na Figura 4. Já na Figura 5 pode-se ver o desempenho por caractere para todos os modelos

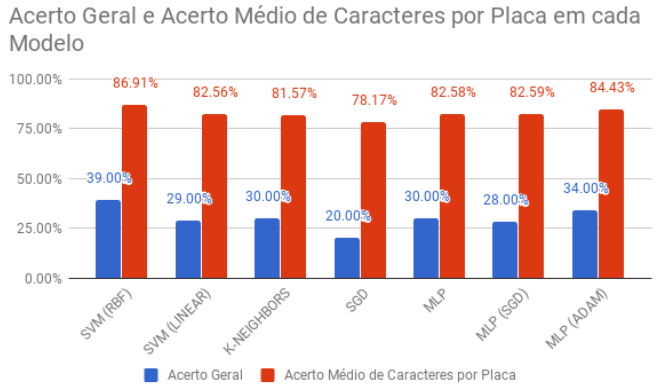

Figura 3. Desempenho de cada modelo em relação a proporção de acertos gerais e acertos médio das strings completas de cada placa (percentagem de caracteres reconhecidos no total da placa)

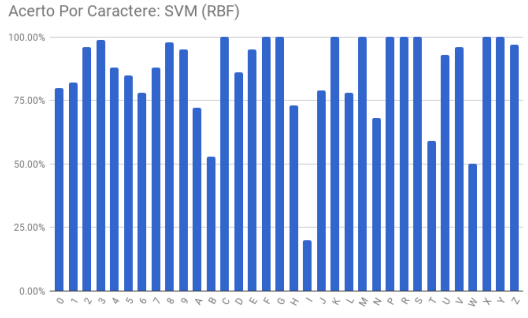

Figura 4. Desempenho do Modelo treinado com SVM com kernel RBF em relação a acerto médio por cada caractere

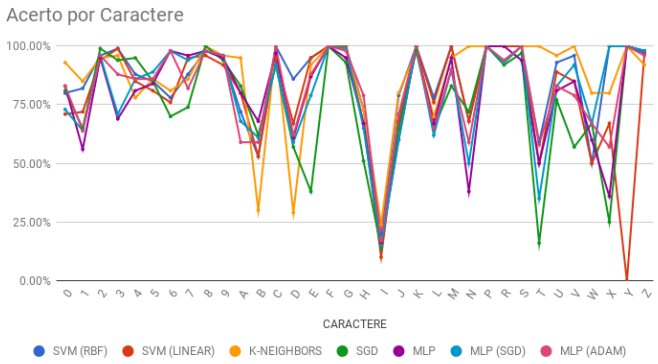

Figura 5. Desempenho de cada modelo em relação ao acerto médio em cada caractere

IV. Conclusão

Com o trabalho desenvolvido foi possível ver o comportamento de cada uma das técnicas utilizando-se poucos dados para treinamento dos modelos, ou seja, analisar quão bom foram os resultados mesmo apresentando poucos exemplos para cada classe, considerando que a quantidade de dados é um dos fatores determinantes para a construção de bons modelos em técnicas de *Machine Learning*.

Comparado com [1] que obtiveram um resultado geral de 93,7% nosso resultado de 58% pode ser considerado baixo, sem considerar que abordagens mais recentes como em [8] os resultados obtidos foram de 99,3%. Entretando, o volume de dados utilizados por estes experimentos foi bem maior que o utilizado no nosso trabalho.

Como trabalho futuro podemos conseguir bases de dados maiores para tentar criar modelos com maiores acurácias e verificar o comportamento de cada uma das técnicas com um volume de dados significativo.

Referências

[1] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 1, pp. 42–53, 2004.

[2] M. Yu and Y. D. Kim, "An approach to korean license plate recognition based on vertical edge matching," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 2975–2980.

[3] J. Kong, X. Liu, Y. Lu, and X. Zhou, "A novel license plate localization method based on textural feature analysis," in *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*. IEEE, 2005, pp. 275–279.

[4] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 3, pp. 377–391, 2008.

[5] K. K. Kim, K. Kim, J. Kim, and H. J. Kim, "Learning-based approach for license plate recognition," in *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, vol. 2. IEEE, 2000, pp. 614–623.

[6] A. Mecocci and C. Tommaso, "Generative models for license plate recognition by using a limited number of training samples," in *Image Processing, 2006 IEEE International Conference on*. IEEE, 2006, pp. 2769–2772.

[7] H. S. S. G. K. A. Ribari, Kalafati, "Detekcija, raspoznavanje i automatski unos registarskih oznaka."

[8] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License plate detection and recognition using deeply learned convolutional neural networks," *arXiv preprint arXiv:1703.07330*, 2017.