Extração de Features de Imagens Reconhecimento e Classificação de Personagens

Tiago Tadeu Madrigar, acadêmico – pg54809 [Universidade Estadual de Maringá | tiago@madrigar.com.br]

Dr. Yandre Maldonado e Gomes da Costa, [Universidade Estadual de Maringá | yandre@din.uem.br]

Abstract. *Este trabalho tem sua motivação a partir de um recurso de mapeamento e identificação utilizada por serviços de streaming e apresenta uma aplicação para segmentação e extração de features de imagens de modo a permitir o reconhecimento e classificação de personagens. O conjunto de imagens utilizado como base para este trabalho foi embasada nos personagens do desenho 'The Simpsons'. O experimento consistiu em utilizar algumas técnicas para extração de features e classificação de imagens, ao final foi possível realizar um estudo comparativo dos resultados obtidos. Para este trabalho o classificador SMO obteve maior índice de instâncias classificadas corretamente.*

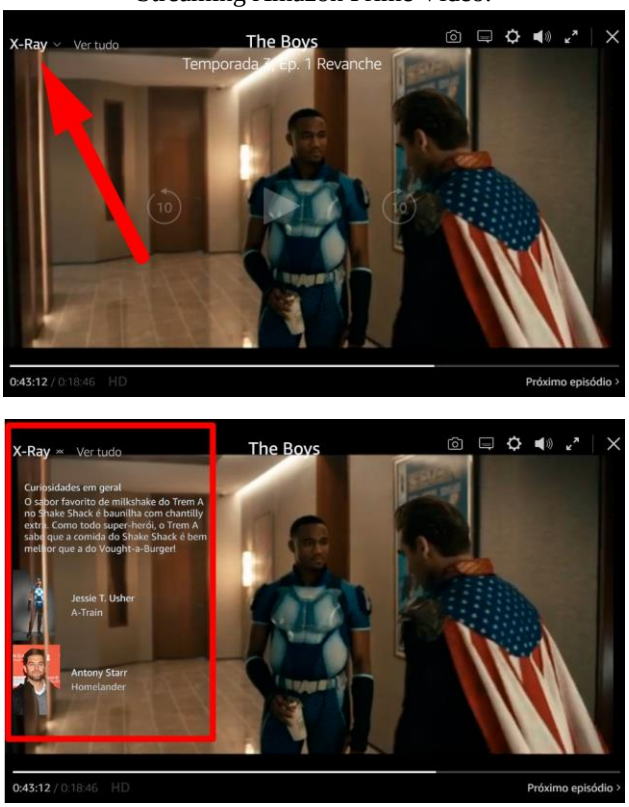
I. INTRODUÇÃO

Uma das evoluções tecnológicas mais marcantes dos últimos anos vem acontecendo no mundo do streaming. Streaming é a tecnologia que permite consumirmos filmes, séries e músicas em qualquer lugar. Em poucos anos, vimos as locadoras de VHS e DVDs se dissolvendo e cada vez mais pessoas estão consumindo entretenimento de maneira mais acessível. Desde seu surgimento até os dias atuais a era do streaming vem se tornando cada dia mais acessível, estável e constante, sendo tendência para os próximos anos que seguem. [2].

A Amazon Prime Video, tem conquistado seu espaço no segmento de serviços de streaming no Brasil. Com um catálogo cada vez mais diversificado, ela conta também com um recurso denominado: X-Ray, que apresenta informações interessantes sobre séries e filmes disponíveis na plataforma, reduzindo a necessidade de buscá-las na internet. Isso foi possível porque a Amazon integrou seu catálogo de filmes e séries ao serviço IMDb (Internet Movie Database), pertencente ao grupo Amazon. Esta plataforma por sua vez, reúne informações completas sobre programas de TV, cinema e música, somado ao serviço de pontuação/classificação média, dada por usuários de todo o mundo.

O recurso X-Ray quando disponível durante a reprodução da série ou filmes apresenta quem são os atores que estão em cena durante aquele momento e a partir daí o usuário pode explorar informações do elenco, como biografias de atores, fotos, filmografias e histórias de fundo dos personagens. No caso de músicas estarem sendo reproduzidas o recurso mostrará o nome da música e até quem a cantou. A Figura 1 apresenta a forma em que o recurso é disponibilizado.

Figura 1 – Recurso X-Ray disponível no serviço de Streaming Amazon Prime Video.



Fonte: O Autor

O recurso X-Ray realiza um processamento digital de imagens em tempo de execução fazendo um rastreamento de acordo com o reconhecimento facial do personagem, fazendo uma pesquisa no banco de

dados do IMDb. Ao identificar o nome do filme ou série, assim como o ator, a função diz exatamente, o nome do personagem que o ator vive no momento, bem como informações derivadas das cenas e curiosidades.

A partir desta motivação, este trabalho, inspirado nos recursos do X-Ray apresenta os resultados da segmentação e extração de características (*features)* de imagens de modo a permitir o reconhecimento de classificação das mesmas [11]. Foram implementadas quatro técnicas distintas de extração de *features*, sendo elas: Raw *features*; Filtro de Gabor; Filtro Sobel e Histograma;

Para comparar o desempenho dessas técnicas foi realizado um experimento com um conjunto de 500 imagens selecionadas de uma base de dados, que após passarem pela fase de extração de *features* foram classificadas a partir de um conjunto de medidas.

E para avaliar a variabilidade do conjunto de dados e a confiabilidade dos modelos treinados com esses dados este trabalho utilizou o método da Validação Cruzada de 10 *folds*, dividindo uma parte para treinamento e as nove partes restantes para teste. Essa metodologia de validação permite verificar se o modelo é suscetível a variações nos dados, permitindo avaliar a confiabilidade das previsões.

II. BASES TEÓRICAS

A visão computacional, como definida, é a ciência responsável pela visão de máquina, a forma como um computador vê o seu entorno, extraindo informações importantes das imagens. Essas informações possibilitam identificar, manipular e pensar sobre os objetos que compõem uma imagem [1].

A partir do momento que a computação pode processar grandes quantidades de dados como imagens, nasceu a área dos sistemas de visão computacional. A visão computacional é o estudo da extração de informação de uma imagem; mais especificamente, é a construção de descrições explícitas e claras dos objetos em uma imagem [1].

Desde a década de sessenta, com o aumento da capacidade de memória e da velocidade dos computadores, alinhadas as técnicas de inteligência artificial o processamento digital de imagens foi impulsionado ganhando bom reconhecimento.

Segundo [15], os sistemas de processamento de imagens devem possuir etapas bem definidas, possuindo a capacidade de extrair apenas informações importantes, onde existir informações irrelevantes. O sistema deve ser capaz de inferir a partir de informações incompletas e ser capaz de reconhecer padrões em imagens o mais independente possível de fatores como posição, tamanho, orientação e mudanças de ponto de vista, ou seja, da independente das transformações geométricas que possam existir.

Para que o reconhecimento de objetos e padrões em imagens possa acontecer de maneira satisfatória, é preciso implementar um conjunto de técnicas de aprendizado de máquina que garanta a extração dessas informações [16]. O Aprendizado de Máquina é uma linha de pesquisa em Inteligência Computacional que estuda métodos capazes de extrair conceitos (conhecimento) a partir de amostras de dados [13].

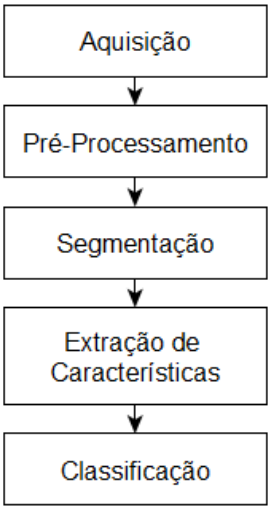
Reconhecer padrões em imagem é uma tarefa que depende da quantidade de *features* que se conhece de cada objeto. Portanto, a extração das informações é uma etapa crucial no processo de reconhecimento [17].

No campo científico, o processamento digital de imagens pode ser entendido como a análise e a manipulação de imagens por computador, envolvendo a identificação e extração das informações presentes para posterior processamento e análise. No modelo de processamento de imagens, existem etapas fundamentais das quais compartilham informações e resultados entre si, produzindo ações que interferem diretamente nas etapas posteriores.

A Figura 2 apresenta as principais etapas do processamento digital de imagens mostrando desde a aquisição onde ocorre a captura e digitalização das informações visuais, até o resultado que se pretende obter.

A primeira etapa é aquisição que pode ser realizada de algum dispositivo. Na etapa do pré-processamento são realizadas tarefas de melhoramento da imagem. A etapa da segmentação tem o objetivo a identificação de objetos ou áreas de interesse, em geral, esta é uma etapa de que pode determinar graus de sucesso ou de eventuais falhas em toda análise [15].

Figura 2 – Principais etapas de um sistema de Processamento Digital de Imagens.



Fonte: Adaptado de [14]

A partir dos objetos identificados na segmentação são extraídas *features* para a etapa de análise ou classificação [14]. Dentro de cada etapa de um sistema de processamento digital de imagens são realizadas tarefas específicas como controle de iluminação (etapa da aquisição), aplicação de filtros (etapa do pré-processamento), identificação de bordas (etapa da segmentação) entre outras [8].

III. MATERIAIS E MÉTODOS

Devido à importância e aos interesses dos cientistas cognitivos, a pesquisa em reconhecimento de faces é tão antiga quanto a própria visão computacional. Sistemas baseados no reconhecimento facial têm ganho grande importância não só na identificação, como no controle de acesso de indivíduos em diversos ambientes [2].

O reconhecimento de faces humanas é um problema bastante complexo para ser implementado de forma automática, devido a diversos fatores como: diferentes variações de orientação e tamanho da imagem, condições de iluminação do ambiente, diferenças na aparência, na expressão facial e na cor da pele, entre outros fatores que influenciam a extração de *features* [20] [17].

Para identificar ou autenticar indivíduos distintos é necessário utilizar *features* biométricas que os diferenciem, tais como partes físicas do corpo, geometria dos membros ou *features* atreladas a impressão digital, sendo esta, amplamente reconhecida e usada desde o final do século XIX. Dentre todas as *features* mencionadas, este trabalho se ocupará o uso da identificação corporal e facial de personagens de série televisiva *The Simpsons*.

Este trabalho utilizou a base de dados denominada: *'The Simpsons Characters Data'*, focada em diversos personagens apresentados durante as vinte e oito temporadas da série. A base de dados é disponibilizada gratuitamente na plataforma Kaggle[1]. O conjunto de dados atualmente possui 43 classes/personagens (uma pasta para cada personagem), divididos com 400 a 2000 imagens cada pasta. A Figura 3 apresenta uma visualização da base de dados utilizada

Figura 3 – Visualização do conjunto de imagens da Base de Dados utilizada.

Fonte: Base de Dados *The Simpsons Characters Data*

Devida a grande quantidade de imagens a serem processadas (cerca de 42 mil), a média das dimensões de cada imagem (640x480), a média dos pontos por polegadas (96 dpi por imagem), o tamanho da base (cerca de 1,13 GB) e devido às limitações computacionais em relação ao hardware (núcleos de processamento e memória), este trabalho precisou limitar o escopo aos cinco personagens/classes principais apresentados na série, sendo estes: *Homer_Simpson, Marge_Simpson, Bart_Simpson, Lisa_Simpson, Maggie_Simpson.*

Devido as inúmeras possibilidades e variações da face, as variedades e modificações dos movimentos corporais e das alterações ambientais da cena que envolvem cores e iluminação, esta aplicação possuirá vários níveis de aplicação e dificuldade.

[1] https://www.kaggle.com/datasets/alexattia/the-simpsons-characters-dataset

A. *Processo da Extração de features*

Todas as funções criadas e utilizadas no processo de extração de *features* deste trabalho foram detalhadas e disponibilizadas no diretório do GitHub do autor[2]. Neste trabalho foi utilizada a linguagem Python, uma linguagem interpretada, composta por diversas bibliotecas, inúmeros módulos para leitura e escrita de arquivos, processamento e análise de imagens digitais [3]. Neste trabalho funções foram criadas para implementar e construir as técnicas para extração de *features*, da base de dados utilizada.

A primeira técnica implementada foi denominada de Raw *features* e possuía algumas *features* distintas. Uma função em Python denominada *createRawFeature* foi criada e sua primeira tarefa, era criar um vetor de *features* tirada da própria imagem sem nenhum tipo de filtro aplicado. A imagem é dividida em uma grade de tamanho onde cada quadrante tem BLOCKS_ROWS x BLOCKS_ROWS de dimensões, então é extraído um vetor comprimido *createVector* para cada quadrante. O tamanho, por tanto é: (*w* / BLOCKS_COLS) * (*h* / BLOCKS_ROWS) * 3 * 3 = (*w* / BLOCKS_COLS) * (*h* / BLOCKS_ROWS) * 9 onde *w* é a largura da imagem e *h* a altura em pixels. Essa técnica foi implementada com intuito de calcular a média dos pixels da imagem com base em sua semelhança de cores, visto que inúmeros personagens da série *The Simpsons* possuem coloração amarela em sua maioria. Outra justificativa para esta implementação desta técnica, é que em particular, imagens de faces, possem várias regiões de intensidade semelhante.

A segunda técnica implementada foi a filtragem com filtros de Gabor. Esse filtro é usado para análise de textura. Através de um conjunto de classes de funções de Gabor é possível representar de forma completa (frequência e orientação) qualquer tipo de imagem. As funções utilizadas nos filtros têm como objetivo extrair atributos para caracterizar diferentes tipos de texturas presentes na imagem. [5]. Usando multicanais texturais, filtros de Gabor permitem a extração de feições, uma vez que os parâmetros principais do filtro como a extensão, a frequência e orientação são determinantes para obter bons resultados. Devido a suas características específicas, através da comparação das respostas de amplitude dos canais é possível verificar e descobrir fronteiras entre as texturas, e dessa forma pode-se separá-las. Os filtros de Gabor podem ser utilizados para outras tarefas de seleção de feições simulando a percepção visual humana, entretendo um dos problemas existentes é o alto custo computacional para compor um banco de filtros completos que representem a imagem original. [9]

Neste trabalho foi implementada uma função em Python denominada *createGaborFeatures*, depois da definição de alguns parâmetros iniciais como a orientação em graus e o tamanho em pixels a função executa uma estrutura de repetição que percorrer uma matriz de filtros com diferentes orientações, e por ser a tarefa com maior custo computacional, o processamento se utiliza da base de imagens onde aconteceu redução da amostra, já mencionada na etapa anterior. A Figura 4 apresenta uma imagem dividida em duas partes, ao lado esquerdo a imagem original, ao lado direito da divisão as texturas evidenciadas pelo filtro de Gabor.

Figura 4 – Aplicação do Filtro de Gabor.

Fonte: Adaptado de [9]

Para Gabor são gerados N vetores correspondente as variações de direção THETA e os comprimentos de onda SIGMA. O resultado do vetor é compactado usando a função *createVector* que reduz o vetor para 3 dimensões. Logo o tamanho total do vetor final de Gabor *g é: n(g)= (θ ∗n (σ) ∗ 3) ∗n (c)*, onde *n(x)* é o tamanho de um conjunto *x* e *c* os canais da imagem. Em python pode ser calculado como: THETA * len(SIGMA) * 3 * NUM_CANAIS

[2] https://github.com/tmadrigar/FeatureExtraction

(1 para imagens em escala de cinza e 3 para coloridas).

A terceira técnica implementada foi a aplicação do filtro Sobel. O filtro Sobel é uma operação voltada a detecção de contornos. O filtro calcula diferenças finitas, dando uma aproximação do gradiente da intensidade dos pixels da imagem, dando a direção da maior variação de claro para escuro e a quantidade de variação nessa direção. O filtro detecta bordas horizontais e verticais separadamente, as cores da imagem são transformadas de RGB para escalas de cinza, o resultado é uma imagem transparente com linhas pretas e alguns restos de cores [13]. A detecção de bordas envolve métodos matemáticos para encontrar pontos em uma imagem onde o brilho das intensidades de pixel muda nitidamente. No caso do sobel, utiliza-se a média, mediana e variância dos pixels resultantes como vetor, desta forma sobel contribui com 3 posições para o vetor final.

Neste trabalho foi implementada uma função em Python denominada *createSobelFeature*, depois carregado o vetor imagens. Inicialmente a função encontra o gradiente da imagem em tons de cinza, o que permite encontrar regiões com arestas nas direções x e y. Na sequência acontece a conversão e aplicação da máscara retornando a imagem com as bordas evidenciadas.

Figura 5 – Aplicação do Filtro Sobel.

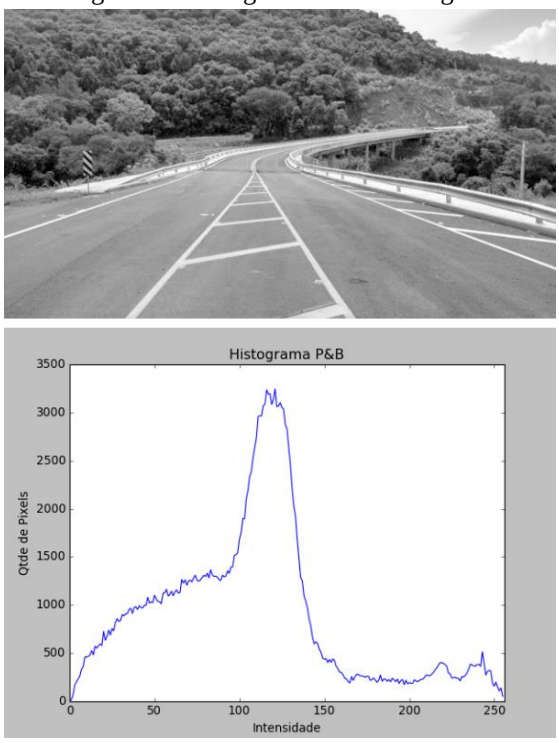
Fonte: Adaptado de [13]

A Figura 5 apresenta uma imagem dividida em duas partes, ao lado esquerdo a imagem original, ao lado direito da divisão as bordas evidenciadas pelo filtro Sobel. No caso do Sobel, utiliza-se a média, mediana e variância dos pixels resultantes como vetor, desta forma Sobel contribui com 3 posições para o vetor final.

A quarta técnica implementada foi a aplicação do Histograma. O histograma de uma imagem é o conjunto de números que indica o percentual de pixels presentes naquela imagem em relação a um determinado nível de cinza ou tom de um determinado canal de cor. a quantidade de pixels mais claros (próximos de 255) e a quantidade de pixels mais escuros (próximos de 0) [5]. A Figura 6 apresenta uma imagem original na parte superior já convertida em escalas em cinza, na parte inferior é apresentado um gráfico exibindo O eixo X com uma distribuição de 0 a 255 que demonstra o valor (intensidade), no eixo Y é exibida a quantidade de pixels daquela intensidade.

Neste trabalho foi implementada uma função em Python denominada *createLBPHistogram*, inicialmente a função carrega as imagens do banco, na sequência inicia o processo de extração de informações para fornecer a representação numérica do conjunto de dados da distribuição. É verificado algumas informações da imagem como tamanho, altura e largura, além dos canais da imagem.

Figura 6 – Histograma de uma imagem.



Fonte: Adaptado de [5]

Uma vez verificada as informações bases da imagem, a função percorre as linhas e colunas e imprimir os diferentes valores de pixel (intensidades) em cada par de linha/coluna. E para melhorar o contraste da imagem é realizada uma equalização, que se dá através da modificação das intensidades de cada pixel. O Tamanho do vetor é o tamanho do histograma que é igual a LBP_NUM+2

IV. RESULTADOS DA CLASSIFICAÇÃO

Em processamento de imagens, o processo da extração de *features* é uma forma especial de redução dimensional. Devido a grande quantidade de dados de entrada e utilizando as técnicas implementadas (extração de *features*), os dados foram transformados em um conjunto reduzido de características melhor representado (também chamado vetor). Dessa forma a final de todo processamento foi gerado um vetor de *features, todos os resultados deste experimento foram disponibilizados no GitHub do autor*[3], o mesmo foi exportado no formato .*CSV*, este vetor possuía todas as *features* extraídas da base de imagens selecionada. Visualizando os dados extraídos em uma planilha foi possível coletar 732 colunas, cada coluna representa uma *features* extraída utilizando as técnicas implementadas pelas funções já descritas na sessão anterior. Devido ao alto custo computacional para o processamento das 43 classes representando os personagens, foi preciso diminuir o escopo focando nos cinco personagens principais da série televisiva. O processo de extração de *features* foi aplicado em uma porção de 100 imagens por classe, totalizando 500 imagens, dessa forma, o vetor de *features* possuía 500 linhas, sendo que cada linha representa as informações extraídas de uma imagem. A Figura 7 apresenta uma parte do vetor de *features* extraído visualizado no formato planilha. Na sequência o vetor de *features* extraído, foi submetido a uma mineração de dados, um processo de busca de padrões consistentes, que se repetem por meio de regras em um grande volume de dados, utilizando técnicas de estatística, inteligência artificial, recuperação da informação, algoritmos de aprendizagem, redes neurais, rede bayesiana entre outras [7].

Figura 7 - Dados extraídos da base de dados

Classes	0	1	2	3	4
bart_simpson	51832167237	7745757858	10604562575	2,17025E+11	1,02245E+14
bart_simpson	28165051446	3164917134	6932419423	8,51312E+13	2,23293E+13
bart_simpson	30624539159	2221926819	13334775009	1,77831E+14	4,65211E+13
bart_simpson	31666333867	2752894859	6576670522	2,4456E+14	8,80226E+13
bart_simpson	54165499406	612563069	17455386093	1,78222E+14	7,03215E+13
bart_simpson	43349914126	4752897641	11091871077	2,21566E+14	8,03785E+13
bart_simpson	28522495889	28713227	5545935539	1,94438E+14	5,06482E+13
bart_simpson	35207377952	298511841	11805484881	2,09227E+14	6,58499E+13
bart_simpson	32475682759	2615441088	5716619505	2,45798E+14	7,66711E+13
bart_simpson	53176726524	5872601221	16434367269	2,3055E+14	8,08405E+13
bart_simpson	31886074406	2661689488	7137362637	2,27692E+14	8,54572E+13
bart_simpson	4634233214	592598602	13153341044	1,02832E+14	4,65053E+13
bart_simpson	53838719662	6550979643	13864838847	1,39057E+14	5,3028E+13
bart_simpson	60942319963	7169225607	20843423162	2,10659E+14	5,78124E+13

Fonte: O autor

Para este processo foi utilizado a ferramenta *Waikato Environment for Analysis* – Weka, um software que oferece diversos algoritmos de mineração de dados que permite a criação de novos algoritmos. Foram utilizados algoritmos disponíveis na ferramenta Weka que estão distribuídos nos seguintes grupos: redes neurais, árvore de decisão, máquinas de vetores de suporte, floresta aleatória, bayesianos e regressão.

Neste experimento os classificadores que tiveram melhores desempenhos foram: baseados em Máquina de vetores de suporte (SVM), e de Otimização Sequencial Mínima (SMO), e o baseado em Árvore de decisão (*RandomForest*).

SVM é uma técnica baseada na teoria de aprendizado estatístico e reúne um conjunto de métodos do aprendizado supervisionado para análise de dados e reconhecimento de padrões. Na etapa de testes, o algoritmo recebe como entrada um conjunto de dados e prediz para qual classe a instância pertence [18]. SMO é um algoritmo iterativo para resolver o problema de otimização dividindo o problema em uma série de subproblemas menores, que são então resolvidos analiticamente [17]. Já o algoritmo *RandomForest*, cria nós para armazenar os caminhos que especificam resultados distintos da decisão. Os nós são responsáveis pelas conferências que irão indicar um caminho ou outro para sequência do fluxo [4].

Na Tabela 1 apresenta alguns resultados do processo de mineração de dados quanto ao vetor de *features* obtido. O classificador SMO conseguiu classificar corretamente 379 imagens de um total de 500, obtendo 75.8% de eficácia quanto a classificação.

[3] https://github.com/tmadrigar/FeatureExtraction

Tabela 1 -Resultado dos classificadores no processo de mineração de dados

Algoritmo	Classificação Correta %	Erro relativo absoluto %	Kappa
SMO	75.8%	37.82%	0.6975
SVM	73.2%	33.5%	0.665
RandomForest	72%	72.23 %	0.65

Fonte: O autor

O classificador não conseguiu classificar corretamente 121 imagens obtendo 24.2% de imprecisão. Quanto a métrica estatística Kappa que permite a precisão de uma classe ou o grau de acerto e concordância, o classificador obteve 0.6975, que interpretando de acordo com tabela especificada pelos autores [12], atingiu concordância substancial. A média do erro relativo absoluto ficou em 0.121 em torno de 37.8207 %, esta medida é utilizada para avaliar a qualidade de um classificador, onde valores menores indicam maior precisão, valores próximos a zero indicam um modelo estatisticamente perfeito.

Para classificar o desempenho de um classificador foi utilizada a matriz de confusão, ela apresenta o número de predições corretas e incorretas em cada classe do problema, permitindo uma análise do comportamento das classes no espaço, uma vez que é possível identificar, dada a classe, em qual classe está o erro apresentado. O número de acertos para cada uma das classes se localiza na diagonal principal da matriz e os demais elementos representam erros na classificação [19]. A Tabela 2 apresenta os resultados do classificador SMO onde a legenda *a* representa a classe *bart_simpson, b* representa a classe *homer_simpson, c* representa a classe *lisa_simpson, d* representa a classe m*aggie_simpson, e* representa a classe *marge_simpson.*

Tabela 2 -Matriz de confusão classificador SMO

a	*b*	*c*	*d*	*e*
70	9	1	11	9
12	68	2	13	5
6	1	92	1	0
9	16	1	66	8
8	2	0	7	83

Fonte: O autor

Observando os resultados apresentados na matriz de confusão foi possível notar que a classe com maior número de acertos para este classificador foi a classe *lisa_simpson* com 92 acertos, já a classe com pior desempenho foi m*aggie_simpson* com 66 acertos.

O classificador SVM conseguiu classificar corretamente 366 imagens de um total de 500, obtendo 73.2% de eficácia quanto a classificação. O algoritmo não conseguiu classificar corretamente 134 imagens obtendo 26.8% de imprecisão. Quanto a métrica estatística Kappa, obteve 0.665, atingindo concordância substancial. A média do erro relativo absoluto ficou em 0.1072 em torno de 33.5%. Para classificar o desempenho do classificador a Tabela 3 apresenta os resultados da matriz de confusão para o classificador SVM.

Tabela 3 -Matriz de confusão classificador SVM

a	*b*	*c*	*d*	*e*
68	8	3	13	8
12	64	4	12	8
4	3	89	2	2
8	12	2	68	10
7	4	1	11	77

Fonte: O autor

Observando os resultados apresentados na matriz de confusão foi possível notar que a classe com maior número de acertos para este classificador foi a classe *lisa_simpson* com 89 acertos, já a classe com pior desempenho foi *homer_simpson* com 64 acertos.

O classificador *RandomForest* conseguiu classificar corretamente 360 imagens de um total de 500, obtendo 72% de eficácia quanto a classificação. O algoritmo não conseguiu classificar corretamente 140 imagens obtendo 28% de imprecisão. Quanto a métrica estatística Kappa, obteve 0.65, atingindo concordância substancial. A média do erro relativo absoluto ficou em 0.2312 em torno de 72.235%. Para classificar o desempenho do classificador a Tabela 5 apresenta os resultados da matriz de confusão para o classificador *RandomForest*.

V. OBSERVAÇÕES FINAIS E TRABALHOS FUTUROS

Observando os resultados apresentados na matriz de confusão foi possível notar que a classe com maior número de acertos para este classificador foi a classe *lisa_simpson* com 89 acertos, já a classe

com pior desempenho foi m*aggie_simpson* com 66 acertos.

Tabela 4 -Matriz de confusão classificador *RandomForest*

a	*b*	*c*	*d*	*e*
64	16	2	8	10
11	69	3	7	10
5	3	89	2	1
9	14	7	60	10
5	8	2	7	78

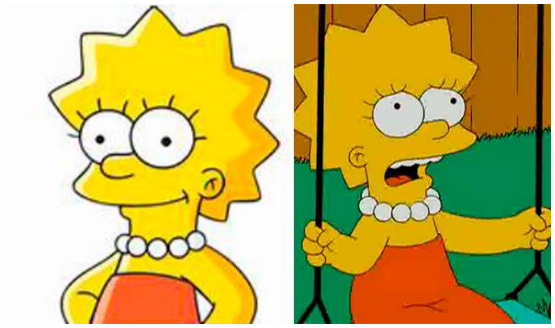
Fonte: O autor

Em todos os classificadores a classe *lisa_simpson* obteve os melhores desempenhos, provavelmente isso se dá ao fato que o formato do cabelo da personagem tenha sido bem caracterizado pelas técnicas de extração de imagens conseguindo resultados bem consistentes, diferenciando-as de outras classes, a Figura 8 apresenta exemplo da classe com bom desempenho.

Outra informação peculiar é que todos os classificadores encontraram em média 10% dificuldades na separação de duas classes em específico sendo, *homer_simpson e bart_simpson* que tiveram resultados imprecisos durante a processo de classificação. A Figura 9 apresenta exemplo das classes conflitantes. É provável que os resultados conflitantes tenham acontecido pelo grau parentesco e similaridades de ambos personagens.

Como trabalhos futuros outras técnicas de extração de *features* poderiam ser implementadas, afim de melhorar os resultados finais e desempenho de cada classificador.

Figura 8 – Exemplo de imagens da classe *lisa_simpson*



Fonte: O autor

Figura 9 – Exemplo de imagens da classe conflitantes

Fonte: O autor

REFERENCIAS

[1] BALLARD, Danna H.; BROWN, Christopher M. Computer Vision. New Jersey: Englewood Cliffs, 1982. 539 p.

[2] CHELLAPPA, Rama; WILSON, Charles L.; SIROHEY, Saad. Human and machine recognition of faces: A survey. Proceedings of the IEEE, v. 83, n. 5, p. 705-741, 1995.

[3] CHITYALA, Ravishankar. Introduction to Python. In: Image Processing and Acquisition using Python. Chapman and Hall/CRC, 2014. p. 39-58.

[4] CUTLER, Adele; CUTLER, D. Richard; STEVENS, John R. Random forests. In: Ensemble machine learning. Springer, Boston, MA, 2012. p. 157-175.

[5] GONZALEZ, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice Hall, New Jersey (2006)

[6] GONZALEZ, Rafael C.; woods, Richard E.; Eddins, S. Digital Image Processing, 2002 by Prentice-Hall. Inc. Upper Saddle River, New Jersey, v. 7458.

[7] HAN, Jiawei; KAMBER, Micheline; MINING, Data. Concepts and techniques. Morgan Kaufmann, v. 340, p. 94104-3205, 2006.

[8] JAHNE, Bernd. Practical handbook on image processing for scientific and technical applications. CRC press, 2004.

[9] JI, Yiming; CHANG, Kai H.; HUNG, Chi-Cheng. Efficient edge detection and object segmentation using Gabor filters. In: Proceedings of the 42nd annual southeast regional conference. 2004. p. 454-459.

[10] JIANG, Xiantao et al. A survey on multi-access edge computing applied to video streaming: some research issues and challenges. IEEE Communications Surveys & Tutorials, v. 23, n. 2, p. 871-903, 2021.

[11] KHODASKAR, A. A.; LADHAKE, S. A. Pattern recognition: advanced development, techniques and application for image retrieval. In: 2014 international

conference on communication and network technologies. Ieee, 2014. P. 74-78.

[12] LANDIS, J. Richard; KOCH, Gary G. An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. Biometrics, p. 363-374, 1977.

[13] MA, Chunxi et al. An improved Sobel algorithm based on median filter. In: 2010 2nd International Conference on Mechanical and Electronics Engineering. IEEE, 2010. p. V1-88-V1-92.

[14] MARQUES FILHO, Ogê; NETO, Hugo Vieira. Processamento digital de imagens. Brasport, 1999.

[15] MITCHELL, Tom M. et al. Machine learning. 1997.

[16] OLSHAUSEN, Bruno A.; FIELD, David J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, v. 381, n. 6583, p. 607-609, 1996.

[17] PLATT, John. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[18] PONTIL, Massimiliano; VERRI, Alessandro. Support vector machines for 3D object recognition. IEEE transactions on pattern analysis and machine intelligence, v. 20, n. 6, p. 637-646, 1998.

[19] STEHMAN, Stephen V. Selecting and interpreting measures of thematic classification accuracy. Remote sensing of Environment, v. 62, n. 1, p. 77-89, 1997.

[20] SUNG, K.-K.; POGGIO, Tomaso. Example-based learning for view-based human face detection. IEEE Transactions on pattern analysis and machine intelligence, v. 20, n. 1, p. 39-51, 1998.