Classificação entre imagens de Cães e Gatos usando algoritmos de Aprendizagem de Máquina em Python

Felipe Archanjo da Cunha Mendes[1]

Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brasil

felipemendes.1999@alunos.utfpr.edu.br

Abstract. *Este artigo apresenta a extração de características de imagens de cães e gatos utilizando a rede neural convolucional VGG16. As características foram armazenadas em conjuntos de dados de treinamento e teste. Em seguida, foram aplicados diferentes modelos de aprendizado de máquina para classificar as imagens. O modelo KNN obteve uma acurácia de 80.7%, o Naive Bayes alcançou uma acurácia de 90.7%, o SVM obteve uma acurácia de 92.9%, a árvore de decisão alcançou uma acurácia de 75.7%, o Random Forest obteve uma acurácia de 92.9%, o Extra Trees alcançou uma acurácia de 92.9%, e o AdaBoost atingiu uma acurácia de 94.4%. Cada modelo teve um desempenho específico na classificação de cães e gatos, com métricas como precisão, recall e F1-score variando para cada classe.*

Resumo. *This article presents the extraction of features from images of dogs and cats using the VGG16 convolutional neural network. The features were stored in training and test datasets. Different machine learning models were then applied to classify the images. The KNN model achieved an accuracy of 80.7%, Naive Bayes achieved an accuracy of 90.7%, SVM obtained an accuracy of 92.9%, decision tree achieved an accuracy of 75.7%, Random Forest obtained an accuracy of 92.9%, Extra Trees achieved an accuracy of 92.9%, and AdaBoost reached an accuracy of 94.4%. Each model had a specific performance in classifying dogs and cats, with metrics such as precision, recall, and F1-score varying for each class.*

1. Introdução

A classificação de imagens é um problema fundamental na área da visão computacional, com aplicações amplas em diversas áreas. Neste artigo, exploramos a classificação de imagens de cães e gatos, utilizando técnicas de aprendizado de máquina. O objetivo é encontrar o algoritmo que obtenha os melhores resultados de classificação para esse problema específico. Para isso, utilizamos a extração de características por meio da rede neural convolucional VGG16, e aplicamos os algoritmos KNN, Naive Bayes, SVM, Árvore de Decisão e Random Forest. Analisamos os resultados obtidos em termos de taxa de acerto e desempenho geral dos modelos.

2. Extração de Características

Neste estudo, as características das imagens de cães e gatos (“Cats and Dogs image classification”, [s.d.]) foram extraídas usando a rede neural convolucional VGG16 (TEAM, [s.d.]), que possui 16 camadas convolucionais e densamente conectadas com pesos pré-treinados. A VGG16 utiliza filtros convolucionais para identificar padrões visuais em diferentes níveis de abstração. Por meio da função "extract_features"

implementada, as imagens foram pré-processadas e as características extraídas e armazenadas em dois arquivos .csv, representando os conjuntos de dados de treinamento e teste, com a primeira coluna indicando a classe de cada amostra.

3. KNN

O modelo KNN (K-Nearest Neighbors) (SCIKIT-LEARN DEVELOPERS, 2019) foi treinado utilizando uma técnica chamada GridSearchCV (SCIKIT-LEARN, 2019), em conjunto com a validação cruzada, para encontrar os melhores parâmetros. O valor da melhor acurácia encontrado foi de aproximadamente 0.786, o que significa que o modelo obteve uma taxa de acerto de cerca de 78.6%. O valor ótimo para o parâmetro K (número de vizinhos considerados) foi de 5, indicando que o modelo considera os cinco vizinhos mais próximos para realizar suas predições. A melhor distância escolhida foi a minkowski, que é uma medida de distância generalizada que engloba outras métricas, como a euclidiana e a manhattan. Por fim, o melhor valor para o parâmetro p, que afeta a fórmula de cálculo da distância minkowski, foi 2, indicando que foi utilizado o cálculo de distância euclidiana.

Com os resultados obtidos no GridSearchCV, um novo modelo KNN foi treinado utilizando os melhores parâmetros encontrados. A acurácia obtida foi de aproximadamente 0.807, indicando que o modelo teve uma taxa de acerto de cerca de 80.7% na classificação das amostras de teste. A tabela de classification report fornece uma visão mais detalhada das métricas de precisão, recall e f1-score para cada classe do conjunto de teste. Para a classe "cats", a precisão foi de 0.81, o recall foi de 0.80 e o f1-score foi de 0.81. Isso significa que o modelo teve um bom desempenho em identificar corretamente os exemplos de "cats". Da mesma forma, para a classe "dogs", a precisão foi de 0.80, o recall foi de 0.81 e o f1-score foi de 0.81, indicando um bom desempenho na classificação de "dogs". A acurácia ponderada e a macro avg foram de 0.81, o que confirma que o modelo foi consistente nas previsões para ambas as classes.

4. Naive Bayes

Similarmente, o modelo Naive Bayes (SCIKIT-LEARN, 2019) foi treinado utilizando a técnica de validação cruzada com o GridSearchCV, em busca dos melhores parâmetros. O valor da melhor acurácia encontrado foi de aproximadamente 0.923, o que significa que o modelo obteve uma taxa de acerto de cerca de 92.3% ao generalizar para dados não vistos. O parâmetro otimizado foi o var_smoothing, que controla a suavização aplicada às estimativas de probabilidade para evitar problemas de divisão por zero. O valor ideal encontrado para esse parâmetro foi de 0.0658.

Com isso, o modelo Naive Bayes foi treinado com os melhores parâmetros encontrados anteriormente e testado com os dados de teste. A acurácia obtida foi de aproximadamente 0.907, indicando que o modelo teve uma taxa de acerto de cerca de 90.7% na classificação das amostras de teste. A tabela de classification report fornece uma visão mais detalhada das métricas de precisão, recall e f1-score para cada classe do conjunto de teste. Para a classe "cats", a precisão foi de 0.98, o recall foi de 0.83 e o f1-score foi de 0.90. Isso significa que o modelo teve um bom desempenho em identificar corretamente os exemplos de "cats", com uma alta precisão. Da mesma forma, para a classe "dogs", a precisão foi de 0.85, o recall foi de 0.99 e o f1-score foi

de 0.91, indicando um bom desempenho na classificação de "dogs". A acurácia ponderada e a macro avg foram de 0.91, o que confirma que o modelo foi consistente nas previsões para ambas as classes.

5. SVM

Da mesma forma, o modelo SVM (Support Vector Machine) (SCIKIT-LEARN DEVELOPERS, 2019) foi treinado utilizando a técnica de validação cruzada com o GridSearchCV. O valor da melhor acurácia encontrado foi de aproximadamente 0.950, o que significa que o modelo obteve uma taxa de acerto de cerca de 95% ao generalizar para dados não vistos. Os parâmetros otimizados foram o C, que controla o custo da classificação errada, com o valor ideal de 0.001, indicando um menor custo de erro, o gamma, que controla a influência de cada exemplo de treinamento, também com o valor ideal de 0.001, indicando uma influência menor, e o kernel, que define o tipo de função utilizada para transformar os dados, com o valor ideal de "linear", indicando a utilização do kernel linear.

Diante disso, o modelo SVM foi treinado com os melhores parâmetros encontrados anteriormente. A acurácia obtida foi de aproximadamente 0.929, indicando que o modelo teve uma taxa de acerto de cerca de 92.9% na classificação das amostras de teste. A tabela de classification report fornece uma visão mais detalhada das métricas de precisão, recall e f1-score para cada classe do conjunto de teste. Para a classe "cats", a precisão foi de 0.98, o recall foi de 0.87 e o f1-score foi de 0.92. Isso significa que o modelo teve um bom desempenho em identificar corretamente os exemplos de "cats", com uma alta precisão. Da mesma forma, para a classe "dogs", a precisão foi de 0.88, o recall foi de 0.99 e o f1-score foi de 0.93, indicando um bom desempenho na classificação de "dogs". A acurácia ponderada e a macro avg foram de 0.93, o que confirma que o modelo foi consistente nas previsões para ambas as classes.

6. Decision Trees

Da mesma forma, o modelo de árvore de decisão (SCIKIT LEARN, 2019) foi treinado. A partir da busca realizada com GridSearchCV, foram identificados os parâmetros que resultaram na melhor acurácia, que foi de aproximadamente 0.880. O critério de divisão utilizado foi a entropia (entropy), que mede a impureza dos nós da árvore. O splitter selecionado foi o random, o que significa que as divisões são escolhidas aleatoriamente entre as melhores disponíveis. O parâmetro max_depth foi definido como 50, o que limita a profundidade máxima da árvore a 50 níveis. O parâmetro min_samples_split foi configurado para 20, o que indica o número mínimo de amostras necessárias para dividir um nó interno. E, por fim, o parâmetro min_samples_leaf foi definido como 5, o que representa o número mínimo de amostras necessárias em um nó folha. Esses parâmetros foram selecionados com base no desempenho do modelo durante a validação cruzada.

Treinando um novo modelo com os parâmetros encontrados, a acurácia obtida foi de aproximadamente 0.757, o que indica que o modelo classificou corretamente cerca de 75.7% das amostras do conjunto de teste. Ao analisar as métricas de precisão (precision), recall e f1-score, podemos observar que o desempenho é razoável. A classe

"cats" possui uma precisão de aproximadamente 0.79, o que significa que 79% das previsões para essa classe estão corretas. O recall para "cats" é de cerca de 0.70, o que indica que o modelo identificou corretamente 70% das amostras da classe "cats". Para a classe "dogs", a precisão é de aproximadamente 0.73, indicando que 73% das previsões para essa classe estão corretas. O recall para "dogs" é de cerca de 0.81, o que significa que o modelo identificou corretamente 81% das amostras da classe "dogs". No geral, a métrica f1-score, que combina precisão e recall, é de aproximadamente 0.76 para ambas as classes.

7. Random Forest

Da mesma forma, o modelo Random Forest (SCIKIT-LEARN, 2018) foi treinado utilizando Grid Search CV com Cross-Validation. A melhor acurácia obtida foi de aproximadamente 0.966, o que indica que o modelo alcançou uma taxa de acerto de cerca de 96.6% no conjunto de treinamento. Os melhores parâmetros encontrados foram: n_estimators=500, max_depth=200, min_samples_split=20 e min_samples_leaf=1. Isso significa que o modelo utiliza 500 árvores de decisão (estimadores) no ensemble do Random Forest. A profundidade máxima de cada árvore é definida como 200. A divisão de um nó interno é permitida se houver pelo menos 20 amostras disponíveis para serem divididas. Além disso, o número mínimo de amostras necessárias para serem consideradas uma folha é 1. Esses parâmetros são considerados os melhores com base na métrica de acurácia e indicam a configuração que resultou no melhor desempenho do modelo Random Forest no conjunto de treinamento.

Com isso foi treinado um novo modelo de Random Forest utilizando os melhores parâmetros encontrados pelo Grid SearchCV. A acurácia obtida foi de aproximadamente 0.929, o que indica que o modelo alcançou uma taxa de acerto de cerca de 92.9% no conjunto de teste. A precisão para a classe "cats" foi de 100%, o que significa que todos os exemplos classificados como "cats" pelo modelo foram corretos. A precisão para a classe "dogs" foi de 88%, indicando que o modelo teve uma taxa de acerto ligeiramente menor para essa classe. O recall para ambas as classes foi de 86% para "cats" e 100% para "dogs", o que mostra que o modelo teve um bom desempenho em recuperar exemplos relevantes para ambas as classes. O f1-score, que é uma média ponderada entre precisão e recall, foi de aproximadamente 0.92 para ambas as classes, indicando um bom equilíbrio entre precisão e recall.

8. Extra Trees

Em seguida foi treinado o modelo Extra Trees (“3.2.4.3.3. sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.22.2 documentation”, [s.d.]) utilizando Grid Search CV com Cross-Validation. A melhor acurácia encontrada foi de aproximadamente 0.968, indicando que o modelo obteve uma taxa de acerto de cerca de 96.8% no conjunto de treinamento. Os melhores parâmetros encontrados foram: n_estimators = 200, max_depth = 200, min_samples_split = 10 e min_samples_leaf = 1. Isso significa que o modelo utilizou 200 estimadores (árvores de decisão) para construir o ensemble, cada uma com uma profundidade máxima de 200. Além disso, o modelo definiu que o número mínimo de amostras necessárias para realizar uma divisão no

treinamento de uma árvore é 10, e o número mínimo de amostras necessárias para formar uma folha é 1. Esses parâmetros são considerados os melhores para o modelo Extra Trees com base na busca realizada.

Com isso, foi treinado o modelo Extra Trees com os melhores parâmetros encontrados através da busca no Grid Search. A acurácia obtida foi de aproximadamente 0.929, indicando que o modelo obteve uma taxa de acerto de cerca de 92.9% no conjunto de teste. Observando as métricas de precisão, recall e F1-score, podemos notar que o modelo obteve um desempenho muito bom para ambas as classes "cats" e "dogs". A classe "cats" teve uma precisão e recall de 1.00 e 0.86, respectivamente, o que significa que o modelo identificou corretamente a maioria dos exemplos dessa classe. Da mesma forma, a classe "dogs" obteve uma precisão e recall de 0.88 e 1.00, respectivamente. A métrica F1-score, que é uma média ponderada da precisão e do recall, também é alta para ambas as classes.

9. Ada Boost

Da mesma forma, foi feita a busca dos melhores parâmetros para o modelo AdaBoost (“sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.22.1 documentation”, [s.d.]) utilizando o método de Grid Search. O melhor conjunto de parâmetros encontrado foi o seguinte: número de estimadores (n_estimators) igual a 500 e taxa de aprendizado (learning_rate) igual a 1.0. A melhor acurácia obtida durante a busca foi de aproximadamente 0.944, indicando que o modelo obteve um desempenho satisfatório na classificação dos dados de treinamento. Esses melhores parâmetros sugerem que o modelo AdaBoost foi capaz de aprender com sucesso um conjunto de estimadores fracos (weak learners) ponderados com uma taxa de aprendizado adequada. Isso resulta em um modelo final mais robusto e preciso na classificação dos dados. O valor do learning_rate igual a 1.0 indica que o peso dos estimadores no conjunto final é considerado igual, o que pode ser uma escolha apropriada dependendo do problema em questão.

Com isso foi treinado um novo modelo AdaBoost com os melhores parâmetros encontrados utilizando o método de Grid Search. A acurácia obtida foi de aproximadamente 0.936, o que indica que o modelo possui um bom desempenho na classificação dos dados de teste. Ao analisar as métricas de precisão, recall e f1-score, é possível observar que o modelo possui um bom equilíbrio entre as classes "cats" e "dogs". A classe "cats" obteve uma precisão de 1.00, ou seja, todos os exemplos classificados como "cats" foram corretamente classificados. A classe "dogs" obteve uma precisão de 0.89, indicando que a maioria dos exemplos classificados como "dogs" também foi corretamente classificada. Ambas as classes apresentaram recall de 0.87 e 1.00, respectivamente, o que significa que o modelo foi capaz de recuperar a maioria dos exemplos relevantes para ambas as classes. O f1-score, que é uma média ponderada da precisão e do recall, foi de 0.93 para a classe "cats" e 0.94 para a classe "dogs". Esses valores indicam um bom equilíbrio entre a precisão e o recall das duas classes.

10. Redes Neurais

Da mesma forma, foi feita a busca pelos melhores parâmetros para redes neurais (“1.17. Neural network models (supervised) — scikit-learn 0.23.1 documentation”, [s.d.]) utilizando o Grid Search CV. A melhor acurácia encontrada foi de aproximadamente 0.951, o que indica um desempenho satisfatório do modelo. Os melhores parâmetros identificados foram: hidden_layer_sizes com (10, 30, 10), activation com 'tanh', solver com 'sgd', alpha com 0.05 e learning_rate com 'adaptive'. Isso significa que a melhor configuração para a camada oculta da rede neural é composta por três camadas com 10, 30 e 10 neurônios, respectivamente. A função de ativação utilizada é a tangente hiperbólica ('tanh'), o solver é o gradiente descendente estocástico ('sgd'), o valor de alpha (taxa de aprendizado) é 0.05 e a taxa de aprendizado é adaptativa. Esses parâmetros foram escolhidos com base no desempenho durante a validação cruzada.

Com isso um novo modelo de redes neurais foi treinado com os melhores parâmetros encontrados durante a busca utilizando Grid Search. A acurácia obtida foi de 0.9, o que indica que o modelo possui uma taxa de acerto de aproximadamente 90% na classificação dos dados de teste. A precision (precisão) para a classe "cats" é de 0.92, o que significa que 92% dos exemplos classificados como "cats" estão corretos. Já para a classe "dogs", a precision é de 0.88, indicando que 88% dos exemplos classificados como "dogs" estão corretos. O recall (revocação) para a classe "cats" é de 0.87, o que significa que o modelo identificou corretamente 87% dos exemplos que realmente eram "cats". Para a classe "dogs", o recall é de 0.93, indicando que o modelo identificou corretamente 93% dos exemplos que realmente eram "dogs". O f1-score é de 0.90 para ambas as classes, o que representa uma medida combinada de precisão e recall.

12. Conclusão

Em resumo, neste artigo foram aplicados diferentes algoritmos de classificação para distinguir entre imagens de cães e gatos. As características das imagens foram extraídas usando a rede neural convolucional VGG16, e dois conjuntos de dados foram criados: um para treinamento e outro para teste. Em seguida, foram utilizados vários algoritmos de classificação, incluindo KNN, Naive Bayes, SVM, Árvores de Decisão, Random Forest, Extra Trees e AdaBoost.

Os resultados mostraram que todos os algoritmos tiveram um desempenho satisfatório na classificação das amostras. O KNN obteve uma taxa de acerto de aproximadamente 80.7%, enquanto o Naive Bayes alcançou uma taxa de acerto de cerca de 90.7%. O SVM teve uma taxa de acerto de aproximadamente 92.9%, e as Árvores de Decisão alcançaram uma taxa de acerto de cerca de 75.7%.

Os algoritmos de ensemble, Random Forest e Extra Trees, alcançaram uma taxa de acerto de aproximadamente 92.9% para ambos. Finalmente, o algoritmo AdaBoost obteve uma taxa de acerto de aproximadamente 94.4%.

Analisando as métricas de precisão, recall e F1-score, observamos que todos os modelos tiveram um desempenho geralmente bom na classificação de ambas as classes, "cats" e "dogs". No entanto, alguns algoritmos se destacaram em determinadas métricas. Por exemplo, o Naive Bayes obteve uma precisão muito alta para a classe "cats", enquanto o SVM e as Árvores de Decisão apresentaram um recall muito alto para a classe "dogs".

Os algoritmos de ensemble, como Random Forest e Extra Trees, mostraram um equilíbrio geral entre precisão e recall para ambas as classes.

Em termos de desempenho geral, o algoritmo que obteve a melhor acurácia foi o AdaBoost, com aproximadamente 94.4%. No entanto, é importante considerar que diferentes métricas podem ser relevantes dependendo do contexto e das necessidades do problema.

Referencias

Cats and Dogs image classification. Disponível em: <https://www.kaggle.com/datasets/samuelcortinhas/cats-and-dogs-image-classification>. Acesso em: 3 jun. 2023.

TEAM, K. Keras documentation: VGG16 and VGG19. Disponível em: <https://keras.io/api/applications/vgg/>.

SCIKIT-LEARN DEVELOPERS. sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.22.1 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>.

SCIKIT-LEARN. sklearn.model_selection.GridSearchCV — scikit-learn 0.22 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>.

SCIKIT-LEARN DEVELOPERS. sklearn.svm.SVC — scikit-learn 0.22 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.

SCIKIT-LEARN. 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.

SCIKIT-LEARN. 1.9. Naive Bayes — scikit-learn 0.21.3 documentation. Disponível em: <https://scikit-learn.org/stable/modules/naive_bayes.html>.

1.17. Neural network models (supervised) — scikit-learn 0.23.1 documentation. Disponível em: <https://scikit-learn.org/stable/modules/neural_networks_supervised.html>.

SCIKIT LEARN. sklearn.tree.DecisionTreeClassifier — scikit-learn 0.22.1 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>.

3.2.4.3.3. sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.22.2 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>.

sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.22.1 documentation. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>.