UNIVERSITY OF PORTO – SCHOOL OF SCIENCES – DEPARTMENT OF COMPUTER SCIENCE

Image Classification - Cats,dogs and butterflies

Amanda Tavares
Arina Sanches
Lirielly Vitorugo
December 18, 2022

1 INTRODUÇÃO

A visão computacional é um ramo da Inteligência Artificial que vem sendo cada vez mais estudado e com novas técnicas e melhorias.

Ao longo deste trabalho serão utilizadas diferentes metodologias de classificação de imagem incluindo redes neurais convolucionais (CNN), *Reinforcement Learning* e técnicas tradicionais de aprendizagem de máquina através de dados tabulares.

Foram testadas diferentes técnicas de *dropout* e *Data Augmentation* para estudar o seu impacto nos resultados obtidos pelos modelos. Além disso, foram estudados diferentes conjuntos de dados para analisar a maneira que os modelos reagiam à inserção de novas classes, testando assim o seu poder de generalização.

2 BACKGROUND

Durante o desenvolvimento deste trabalho recorreu-se a diferentes técnica de inteligência artificial, nesta secção vamos explanar sobre as mais relevantes para o presente trabalho.

Dentre elas, a visão computacional tem tido grande relevância porque elas nos permite extrair informações de imagens, vídeos e outras entradas visuais e tomar decisões ou fazer recomendações. Fazendo um paralelo, se a IA permite que os computadores pensem, então a visão computacional permite que ela veja, observe e compreenda. É uma área que ainda lida com muitos desafios, reconhecer padrões que para nós humanos podem ser triviais, não são facilmente reconhecidos por uma inteligência artificial. Humanos tem a vantagem de ter uma

vida inteira de contexto, que ensinam o nosso cérebro a diferenciar imagens, ter noções de distancia, movimento e conseguir identifica-las mesmo na presença de ruído.

A descoberta das redes neurais convolucionais (CNN), representou um grande avanço para a área de visão computacional. Elas permitem a extração de conteúdo de cada vez mais alto nível das imagens. As CNN recebem como entrada os dados de pixel brutos de imagens e consegue aprender a partir disso como extrair variáveis, eliminando assim a necessidade de pré-processamento das imagens para extrair variáveis como textura e formas.

Outras técnicas vem sendo exploradas para melhorar o desempenho de modelos de classificação de imagens, entre elas o *Reinforcement Learning*. Esta é uma área do aprendizagem de máquina, em que um agente inteligente, definido por um programa de computador, interage com o ambiente e aprende a agir de forma a maximizar a noção de recompensa cumulativa.

Reinforcement Learning é um dos 3 paradigmas da aprendizagem de máquina, ao lado da aprendizagem supervisionada e não supervisionada. Diferentemente da aprendizagem supervisionada que precisa que as observações possuam rótulos e de ações para serem explicitamente corrigidos. O mais comum é que o ambiente esteja em um formato de processo de decisão de Markov (MPD), porque muitos algoritmos de *Reinforcement Learning* utilizam técnicas de programação dinâmica. Um MPD é um modelo matemático para tomada de decisão em situações em que os resultados são parcialmente aleatórios e parcialmente sob controle de um tomador de decisão. Ele consiste em um conjunto de estados, um conjunto de ações, uma função de transição e uma função de recompensa.

A principal diferença em relação aos métodos clássicos de programação dinâmica, é que o *Reinforcement Learning* não assume que seja conhecido um modelo matemático especifico do MPD e que o objetivo é ter MPDs grandes onde os métodos se tornam inviáveis.

Além de utilizar as imagens, também podemos utilizar outras estruturas de dados, como os tabulares, que nos permite aplicar outros modelos de aprendizagem de máquina. Este tipo de dados permite que o modelo faça a classificação através de características que só são perceptíveis a olho humano, estas características permitem diferenciar mais facilmente uma classe da outra, como é o caso neste trabalho, permite distinguir características específicas de certos animais.

3 Materiais e Métodos

3.1 Dados e Algoritmos

O conjunto de dados foi extraído do Kaggle, contém 25000 imagens de cães e gatos, tendo o mesmo número de observações para cada classe, sendo que 25% destas imagens foram utilizadas para teste e 75% para treino.

Durante a elaboração deste trabalho foram utilizados diferentes modelos de redes neurais convolucionais profundas (CNN), dentre eles VGG e EfficientNet.

VGG representa a sigla do termo em inglês *Visual Geometry Group*, que é uma das arquiteturas de reconhecimento de imagem mais populares atualmente. É a arquitetura padrão de Rede Neural Convolucional (CNN) profunda, com múltiplas camadas, foi baseada na análise de como aumentar a profundidade destas redes. São utilizados pequenos filtros de 3x3 pixels. Essas redes se caracterizam pela sua simplicidade, sendo que as outras componentes são

camadas de *pooling* e a camada completamente conectadas. A profundidade se refere ao número de camadas que cada VGG possui, sendo por exemplo VGG-16 um modelo com 16 camadas convolucionais. A arquitetura VGG é a base dos modelos de reconhecimento de objetos,

Outra arquitetura de rede neural convolucional é a EfficientNet, que também é um método de dimensionamento que dimensiona uniformemente todas as dimensões de profundidade/largura/resolução usando um coeficiente composto. Diferentemente dos métodos convencionais, que dimensionam esses fatores arbitrariamente, o método de dimensionamento aplicado pela EfficientNET dimensiona uniformemente a largura, a profundidade e a resolução da rede com um conjunto de coeficientes de dimensionamento fixos.

Transfer learning é um método de aprendizagem de máquina onde o modelo desenvolvido para uma tarefa, é reutilizado como ponto de partida para um modelo de uma segunda tarefa. É uma abordagem muito popular na área de aprendizagem profunda, onde modelos pré-treinados são utilizados como um ponto de partida para visão computacional e processamento de linguagem natural, considerando que para desenvolver um modelo de rede neural para estas tarefas, necessita de muitos recursos computacionais e tempo de execução.

Este modelo permite que estes tipos de problemas sejam resolvidos com maior eficiência. Na visão computacional, por exemplo, as redes neurais geralmente tentam detetar arestas nas camadas anteriores, formas na camada intermediária e alguns recursos específicos de tarefas nas camadas posteriores. No *transfer learning*, as camadas iniciais e intermediárias são usadas e apenas re-treinar as últimas camadas.

Treinar redes neurais profundas com conjuntos de dados maiores, pode resultar em um modelo com melhor capacidade de generalização, técnicas de *augmentation* podem ser utilizadas para criar variações das imagens do conjunto de dados original. *Data augmentation* aplicado em imagens é uma técnica que pode ser utilizada para expandir artificialmente o tamanho do conjunto de dados de treino, através da criação de versões modificadas da imagens pertencentes ao conjunto de dados. O aumento de dados também pode atuar como uma técnica de regularização, adicionando ruído aos dados de treinamento e incentivando o modelo a aprender as mesmas variáveis, independentemente de sua posição na entrada.

3.2 Classificação utilizando imagens

O trabalho teve início com a classificação de imagens, utilizando o conjunto de dados de imagens de cães e gatos. As imagens contidas no conjunto de dados tinham diferentes tamanhos, por isso optou-se por redimensionar todas para o mesmo tamanho, no caso 200x200 pixels, de forma a tornar os dados mais uniformes de forma a tornar o treino do modelo mais rápido.

Numa primeira etapa, exploramos diferentes arquiteturas da VGG, de forma a encontrar a que melhor se adaptasse ao conjunto de dados e consequentemente obtivesse um melhor desempenho na classificação das imagens. Para obter os diferentes modelos de VGG, são empilhadas camadas convulsionais com filtros pequenos de 3x3 e camadas de *max pooling*. O *padding* é utilizado nas camadas convolucionais para garantir que a altura e largura do mapa de variáveis do resultado são compatíveis com os da entrada. As camadas convolucionais e de *padding* formam um bloco, este pode ser repetido múltiplas vezes, a cada repetição o

número de filtros é sucessivamente aumentado, sendo este iniciado com 32 e dobrado a cada repetição. Os modelos contam ainda com uma camada de *Flatten* e uma camada totalmente conectada. A função de ativação *ReLU* foi a utilizada em todas as camadas, exceto na camada de saída em que foi utilizada a *sigmoid*.

Foram então testadas 3 versões da VGG com o conjunto de dados inicial de gatos e cães. Sendo elas, a VGG-1 que possui apenas um bloco da combinação camada convulsiona e *max pooling*, a VGG-2 que possui 2 blocos e a VGG-3 que possui 3 blocos. O VGG-3 foi escolhida como o modelo *baseline*.

As redes neurais podem facilmente gerar *overfit* quando existem poucas amostras no conjunto de dados. Uma das técnicas disponíveis para lidar com este problema e também diminuir o erro de uma rede neural profunda é a regularização utilizando o *dropout*. O *Dropout* permite a simulação de um grande número de diferentes arquiteturas de rede, utilizando uma única rede e com baixos custos computacionais. Em razão disso foi aplicada a regularização utilizando o *dropout*. Tendo sido adicionado um *dropout* de 20% após cada um dos bloco da VGG-3 e um *dropout* de 50% após a camada totalmente conectada.

Em seguida foi realizada a técnica de *Image Augmentation*, as imagens do conjunto de treino foram aumentadas com um pequeno deslocamentos aleatórios de 20% na horizontal e na vertical, assim como inversões horizontais aleatórias que criam uma imagem espelhada de uma imagem. O novo conjunto de dados gerado foi então utilizado para treinar o modelo VGG-3.

O próximo modelo testado foi o VGG-16, que possui 16 camadas, sendo este dividido em duas partes principais, a parte responsável por extração de variáveis que é constituída por blocos VGG e a parte do classificador que é constituída por um conjunto de camadas totalmente conectadas e a camada de saída. Para o nosso problema mantemos a parte de extração de variáveis e treinamos a parte do classificador para que ele possa se adaptar ao conjunto de dados dos cães e gatos. Para isso é necessário remover as camadas totalmente conectadas da extremidade de saída do modelo e adicionar novas camadas totalmente conectadas para interpretar o modelo e realizar predições.

O modelo final escolhido foi o VGG-16, por ter obtido o melhor desempenho entre os modelos testados. Usualmente são utilizados os conjuntos de dados de treino e teste para treinar o modelo final, entretanto os conjuntos de dados de teste não possuem categorias, pelo que foram utilizados apenas o conjunto de dados de treino. O modelo foi salvo num arquivo, para que possa ser utilizado em previsões futuras.

Terminado o ciclo de treinos e testes com o conjunto de dados originais, passou-se então para a etapa seguinte que consistiu em aumentar o conjunto de dados, através da inserção de novas imagens com ruído branco e retângulos posicionados aleatoriamente de forma a cobrir partes das imagens. Foram adicionadas mais mil imagens de cães e mil de gatos aplicando ruído branco e duas mil de cada com retângulos. Todos os passos que foram aplicados no conjunto de dados original, foram repetidos para este novo conjunto de dados.

Para fins comparativos, testamos um modelo diferente de *Transfer learning*. O modelo escolhido foi o EfficientNetB7, apresentado anteriormente, por ser o que apresentava o melhor resultado no Keras. Para esta etapa, foi utilizado o conjunto de dados de gatos e cães com *Image Augmentation* gerado na etapa anterior.

Por fim, adicionamos uma nova classe de animal ao conjunto de dados original, o animal

escolhido foi Borboleta. Este conjunto de dados foi tirado do Kaggle, chama-se "2022 Dataset of Butterfly Mimics". Obtivemos um conjunto de dados desbalanceado com 12500 cães, 12500 gatos e 3253 borboletas. Repetimos então todo o ciclo de treino e testes que tínhamos feito com o conjunto de dados original, adicionando também o EfficientNetB7 ao conjunto de modelos testados.

Inicialmente tínhamos notado o problema de *data leakage* presente no tutorial, uma vez que ele utiliza o mesmo conjunto de dados para teste e validação, Porém, quando foi avisado que poderíamos mudar o tutorial, pelas limitações do hardware não conseguiríamos rodar todos os experimentos novamente, devido ao tempo de execução. Optou-se então rodar novamente os experimentos apenas para os modelos, VGG-16 e o EfficientNetB7 e comparar os resultados com os obtidos com o *data leakage*. Para isso, criamos para além do conjunto de validação, para além dos de treino e teste. Utilizamos 20% dos dados para teste, e os 80% restantes, dividimos 90% para treino e 10% para validação.

3.3 Classificação utilizando dados tabulares

Após realizar a classificação utilizando o conjunto de imagens, exploramos a realização da mesma tarefa, porém utilizando dados tabulares. Procuramos no Kaggle um conjunto de dados sobre cães, gatos e borboletas que possuísse as descrições das suas características. Para que conseguíssemos treinar um modelo e comparar os resultados obtidos, com o resultado dos modelos obtidos a partir do conjunto de dados inicial de imagens. Porém não conseguimos encontrar um conjunto de dados que se adequasse ao nosso problema. Decidimos escolher aleatoriamente 50 imagens de cada uma das classes em estudo e a partir delas, gerar um conjunto de dados com as descrições das mesmas. As variáveis escolhidas foram:

- size of nose
- shape of ears
- hair
- wings
- size
- antenna

Procuramos escolher variáveis que ajudassem o modelo a mais facilmente distinguir entre as classes de animais. Por exemplo apenas as borboletas possuem asas e antenas e não possuem pelo. Identificamos também que o formato da orelha e o porte do animal em conjunto distinguem bem os cães dos gatos.

Escolhemos o *Multi-layer Perceptron* do sklearn para realizar a classificação. Começamos por criar um modelo utilizando apenas os dados referentes a cães e gatos. O conjunto de dados foi divido em 70% para treino e 30% para teste. De seguida criamos um novo modelo utilizando o conjunto de dados completo com cães, gatos e borboletas. Mantivemos a mesma divisão para treino e teste.

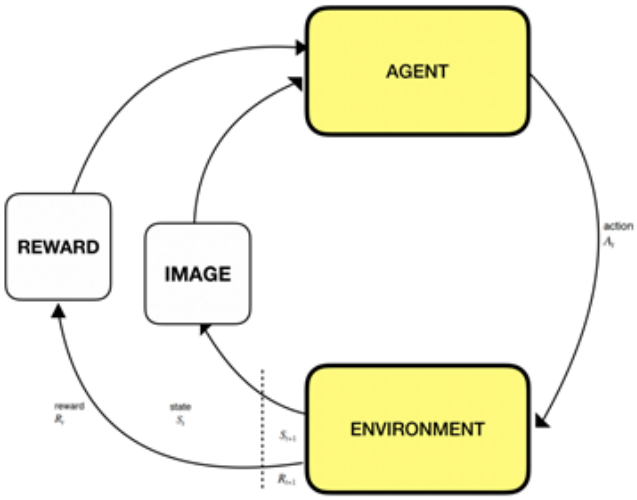


Figure 3.1: Reinforcement Learning.

3.4 CLASSIFICAÇÃO UTILIZANDO DADOS *Reinforcement Learning*

Além das formas de classificação apresentadas acima, também utilizamos um algoritmo de *Reinforcement Learning* para criar um classificador de imagens. Nesta tarefa foi utilizada o conjunto de dados original, porém por limitações impostas pelo hardware, conseguimos utilizar apenas 1000 das 25000 observações presentes no conjunto de dados. A figura 3.1 ilustra de forma simplificada como o processo de aprendizagem funciona. O agente vai interagir com o ambiente através de uma acção. Ele então vai definir uma categoria para aquela observação, os estados, e então recebe uma recompensa que pode ser positiva ou negativa, dependendo dele ter acertado ou errado a categoria da imagem.

Agora iremos explicar alguns dos elementos presentes no processo de aprendizagem utilizando *Reinforcement Learning* para classificação de imagens.

- **State** - Um dados estado S no nosso ambiente são os dados do conjunto de treino. No problema em estudo seria um conjunto das imagens.
- **Action** - A ação alpha do agente é o rótulo da amostra de treinamento. O problema em estudo é um problema binário, onde o agente só pode escolher entre o conjunto de ações A = 0,1, onde 0 é a classe gato e 1 é a classe cachorro.
- **Reward** - A recompensa r é o *feedback* que o ambiente devolve ao agente para que ele meça seu sucesso em classificar o estado s corretamente.
- **Discount factor** - O fator f $\in [0,1]$, pesa na importância das recompensas futuras. Como estamos trabalhando na classificação de imagens, amostras consecutivas não são correlacionadas e cada imagem precisa ser classificada corretamente. Portanto, um valor baixo para f seria uma escolha melhor.

- **Exploration rate** - A taxa e $\in [0,1]$, quando é definida como 1, significa que as ações tomadas são puramente baseadas na exploração, por outro lado, se o valor for 0 as ações tomadas são uma exploração do conhecimento do agente.
- **Episode** - O episódio e termina quando todas as amostras de treinamento no conjunto de treinamento forem passadas para o agente classificar.

O objetivo do agente é maximizar a recompensa cumulativa ao longo do tempo. Para isso, o agente deve aprender uma política, que é um mapeamento de estados para ações que diz ao agente qual ação tomar em cada estado. O agente pode aprender a política ótima através de tentativa e erro, usando um algoritmo de aprendizagem como Q-learning ou SARSA.

Em geral, a representação de um problema de *Reinforcement Learning* e a função de recompensa dependem do problema específico a ser resolvido. O agente, o ambiente, os estados, as ações, a função de transição e a função de recompensa devem ser todos definidos para que o problema possa ser modelado como um PDM e resolvido usando aprendizado por reforço.

Para aplicar esta estratégia, utilizamos uma biblioteca do python chamada IMBDRL, ela implementa esta estratégia em conjunto com uma Rede neural convolucional simples. Mais uma vez, tivemos problemas por causa das limitações do *hardware*, tivemos que redimensionar alguns dos parâmetros, para que conseguíssemos rodar o algoritmo esgotar os recursos disponíveis no computador.

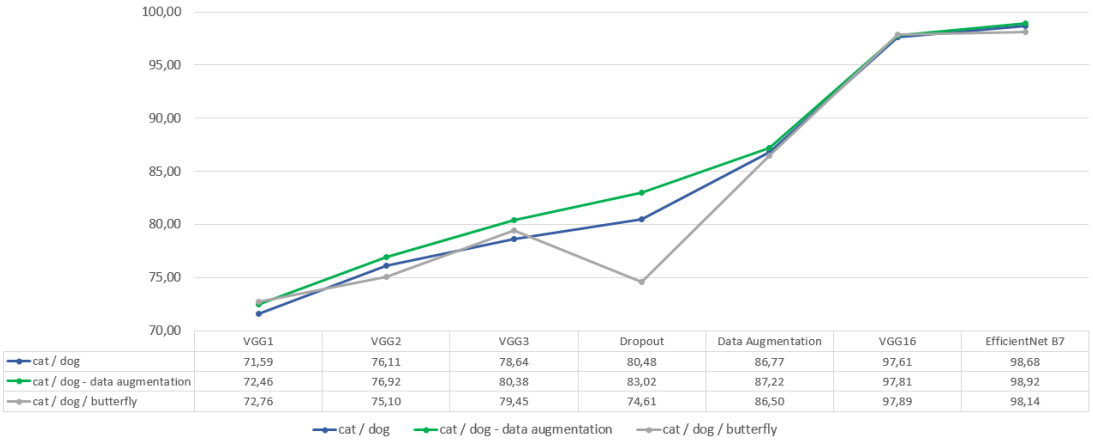
4 Resultados e Discussões

4.1 Classificação utilizando imagens

Na figura4.1 apresentamos os resultados obtidos com os modelos VGG-1, VGG-2, VGG-3, VGG-16, EfficienteNet B7 e aplicando *dropout* e *data Augmentation* juntamente com o modelo VGG-3. Cada linha do conjunto de dados representa os resultados utilizando um conjunto de dados diferentes, a linha azul é referente ao primeiro conjunto de dados testado, que continha apenas gatos e cães, a linha verde representa o conjunto de dados gerado com o conjunto de dados original e adicionando novas imagens através de *data Augmentation* com a inserção de ruído branco e retângulos, por fim a cinza representa o conjunto formado pelas imagens originais de cães e gatos, adicionando as borboletas.

Podemos identificar uma tendência crescente nos resultados obtidos pelos modelos, entretanto podemos realçar um pico quando utilizamos os modelos VGG-16 e EfficientNet B7, com esses modelos obtivemos uma melhora significativa na acurácia em todos os conjuntos de dados. Esses dois modelos foram os que obtiveram o melhor resultado utilizando os 3 conjuntos de dados em estudo. Também é notável uma queda na acurácia ao utilizarmos o *dropout* no conjunto de dados que contém as borboletas, o que já podia se esperar visto que o *dropout* não funciona tão bem em conjuntos de dados desbalanceados porque ele pode "desligar" neurônios que são importantes para a classificação de exemplos minoritários.

O último experimento realizado foi utilizando o conjunto de dados contendo gatos, cães e borboletas e o modelo VGG-16. Porém desta vez utilizando um conjunto de validação, para evitar o *data leakage*. Como já era de se esperar os resultados pioram, entretanto não esperávamos uma piora tão significativa.



	VGG1	VGG2	VGG3	Dropout	Data Augmentation	VGG16	EfficientNet B7
cat / dog	71,59	76,11	78,64	80,48	86,77	97,61	98,68
cat / dog - data augmentation	72,46	76,92	80,38	83,02	87,22	97,81	98,92
cat / dog / butterfly	72,76	75,10	79,45	74,61	86,50	97,89	98,14

Figure 4.1: Acurácia dos modelos criado a partir das imagens

O valor da acurácia que antes tinha sido de 97.89, passou para 43.73, as outras métricas também obtiveram resultados baixos, tendo alcançado *Recall* de 43.73%, *Precision* de 47.25% e F1 de 27.29%.

Ter classes não balanceadas para os humanos não é um problema, os humanos são capazes de distinguir animais mesmo tendo visto muito mais uns que outros, como no exemplo que está a ser estudado, entretanto para o modelo, como já foi possível notar, isto pode se tornar algo que dificulta a classificação e enviesa o modelo.

4.2 Classificação utilizando dados tabulares

Os resultados obtidos utilizando *Multi-layer Perceptron* foram satisfatórios, como já era esperado, uma vez que as características presentes no conjunto de dados diferenciavam bem as classes. Os dois modelos, conseguiram resultados superiores a maioria dos obtidos pelos modelos treinados com imagens, na etapa anterior. Na figura 4.2, apresentamos os valores de acurácia obtidos pelo MLP ao ser treinado com o conjunto de dados contendo cães e gatos e também com o conjunto contendo cães, gatos e borboletas.

Analisando a primeira matriz de confusão apresentada na figura **??**, percebemos que o modelo consegue identificar bem as duas classes, porém acaba errando mais cães do que gatos. Ao adicionar as borboletas ao conjunto de dados, o modelo conseguiu um melhor resultado, podemos ver isso através da acurácia que subiu de 90% para 98% e da segunda matriz de confusão onde constatamos que ele errou a classificação de apenas uma observação.

4.3 Classificação utilizando dados *Reinforcement Learning*

Os resultados obtidos utilizando *Reinforcement Learning*, apresentados na figura 4.4 não foram tão bons como esperado por conta de limitações de *hardware*. Os resultados de F1,

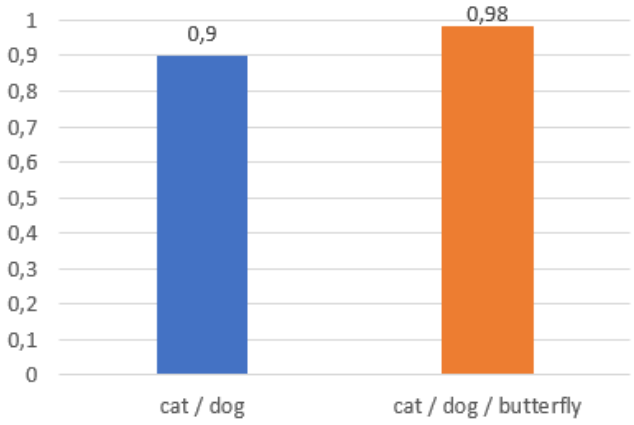


Figure 4.2: Acurácia dos modelos com dados tabulares

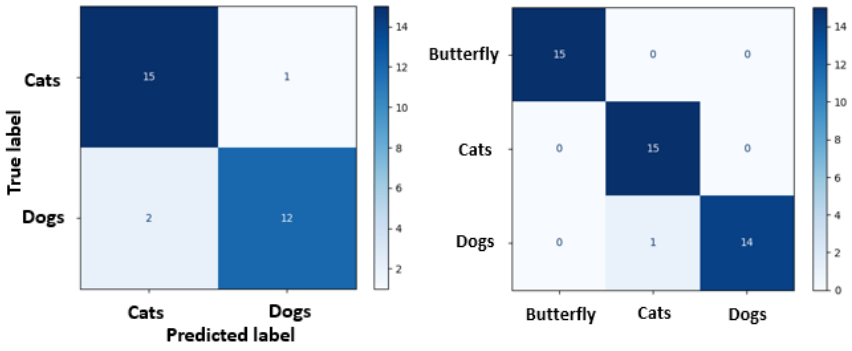


Figure 4.3: Matrizes de confusão dos modelos com dados tabulares.

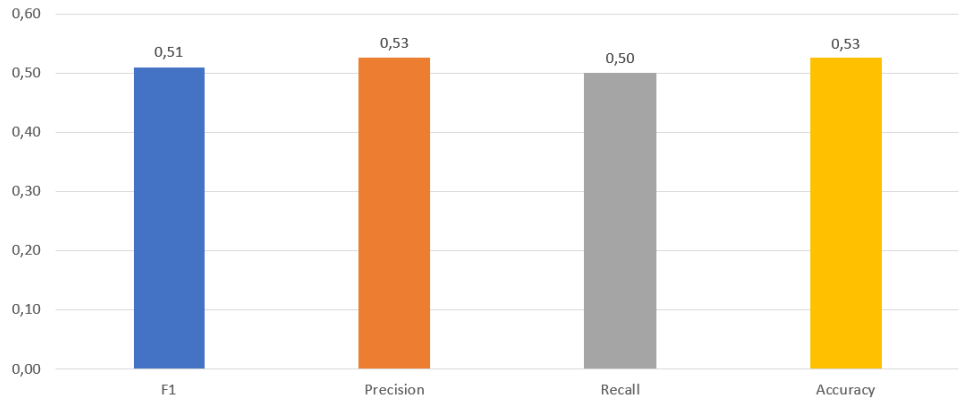


Figure 4.4: Métricas do modelo de Reinforcement Learning.

Precisão, Recall e Acurácia ficaram em torno de 50%, o que se aproxima de um resultado obtido a partir de classificações aleatórias visto que existem apenas duas classes.

Após pesquisas sobre o assunto na literatura, foi possível constatar que este tipo de metodologia pode não ser a mais eficaz em problemas de classificação de imagens por ser muito custosa computacionalmente e demorar muito tempo para convergir.

5 Conclusão

Os algoritmos de *Transfer Learning* foram os que obtiveram melhores resultados na classificação de imagens utilizando o conjunto de dados formado por imagens. Utilizando os dados tabulares obtiveram resultados tão bons quanto (?) os modelos utilizando imagens. Porém este método é menos custoso computacionalmente que as CNNs, mas na prática é difícil encontrar um conjunto de dados que possua esse tipo de anotações. A estratégia utilizada com *Reinforcement Learning* é muito custosa computacionalmente e neste projeto, não gerou resultados satisfatórios.

Fazendo um paralelo entre *Reinforcement Learning* e a forma como uma criança aprende, pode-se dizer que querer que um algoritmo aprenda como uma criança tende a ser um objetivo ambicioso, uma vez que as crianças são capazes de aprender de forma rápida e flexível em um amplo conjunto de tarefas.

No entanto, podem ser citadas algumas características que um modelo de *Reinforcement Learning* ideal para uma tarefa de classificação de imagem poderia ter:

- Capacidade de processar grandes quantidades de dados: um modelo de *Reinforcement Learning* precisaria de muitos exemplos de imagens para aprender a classificá-las corretamente, assim como uma criança precisa de muitas exemplos para aprender a reconhecer objetos.

- Flexibilidade na escolha de ações: um modelo ideal deveria ser capaz de realizar várias ações diferentes, como rotacionar uma imagem, aplicar filtros ou realizar segmentação,

a fim de melhorar a precisão da classificação.

- Capacidade de aprender de forma autônoma: o modelo deveria ser capaz de aprender por conta própria, sem depender de supervisão constante.
- Capacidade de adaptar-se a novas tarefas: o modelo deveria ser capaz de adaptar-se a novas tarefas de classificação sem precisar ser totalmente re-treinado.
- Capacidade de aprender a partir de *feedback* negativo: o modelo deveria ser capaz de aprender com base em erros e *feedback* negativo, assim como uma criança pode aprender com base em correções.

No entanto, é importante lembrar que esses objetivos são bastante ambiciosos e podem ser difíceis de alcançar com os modelos atuais de *Reinforcement Learning*.

Como sugestão de trabalho futuro seria importante utilizar um dispositivo com maior capacidade para alterar os parâmetros do modelo com *Reinforcement Learning*, conseguir utilizar uma amostra maior e assim, possivelmente, obter resultados melhores.

6 BIBLIOGRAFIA

https://www.ibm.com/topics/computer-vision
https://paperswithcode.com/method/vgg
https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/
https://paperswithcode.com/method/efficientnet
https://machinelearningmastery.com/transfer-learning-for-deep-learning/
https://builtin.com/data-science/transfer-learning
https://www.kaggle.com/datasets/fca804bc32fc65614ee308bd728b530e36ca3d3cbb7c770da10a975da6b8337e
https://www.kaggle.com/datasets/antoreepjana/animals-detection-images-dataset