Classificação de Imagens de Esportes Utilizando Redes Neurais Convolucionais

Felipe Fonseca G. Neiva, *UFG*

***Abstract*—Este trabalho explora a classificação de imagens de 100 esportes distintos utilizando redes neurais convolucionais. Dois modelos foram implementados: o Modelo 1, que emprega uma arquitetura convolucional tradicional, e o Modelo 2, que incorpora camadas de concatenações paralelas, similar à arquitetura GoogleLeNet. Os resultados, apresentados por meio de gráficos e tabelas, destacam métricas de acurácia, precisão e perda nos conjuntos de treino e teste. O Modelo 1 atingiu uma acurácia de 92,20%, enquanto o Modelo 2 alcançou 98,40% no conjunto de validação. Técnicas de data augmentation, como giros horizontais e verticais, rotações e ajustes de brilho e contraste, foram aplicadas para aumentar a robustez dos modelos. O trabalho incluiu a visualização de amostras do conjunto de teste com as predições de classe e as classes reais, proporcionando uma visualização do desempenho dos modelos.**

***Index Terms*—Redes Neurais Convolucionais, Data Augmentation, Classificação de Imagens, Deep Learning, Inteligência Artificial.**

I. Introdução

A CRESCENTE disponibilidade de dados e os avanços no campo do aprendizado profundo têm sido catalisadores para o desenvolvimento de modelos cada vez mais sofisticados de reconhecimento de imagens em diversos domínios. Diante desse cenário, este trabalho propõe uma análise por meio da implementação e avaliação de dois modelos de redes neurais convolucionais (CNNs) destinados à classificação de imagens representativas de uma ampla gama de esportes.

O objetivo central é construir modelos capazes de identificar o esporte retratado em cada imagem de um dataset composto por representações de 100 esportes distintos. O dadtaset é composto por 14.493 imagens no total, sendo destinadas 500 para validação, 500 para testes e o resto para treino. Alguns exemplos de imagens do dataset com suas respectivas classes são mostrados na Figura 1.

Mais do que meramente desenvolver sistemas eficazes de reconhecimento de imagens, este estudo visa aprofundar a compreensão sobre como diferentes abordagens e técnicas influenciam a qualidade e o desempenho desses modelos, aplicando técnicas de regularização e estratégias de data augmentation.

II. Metodologia

O PROCESSO de desenvolvimento dos dois modelos foi guiado por uma abordagem iterativa e experimental. Inicialmente, explorou-se uma variedade de configurações de

Fig. 1: Exemplo de imagens do dataset

hiperparâmetros, como taxa de aprendizado, tamanho dos batchs, decaimento de pesos, taxa de droupout e taxa de diminuição da taxa de aprendizado para cada um dos modelos.

Ademais, aplicou-se uma série de transformações de data augmentation, de forma a aumentar a diversidade e contribuir para que o modelo generalizasse melhor para dados não vistos durante o treinamento, reduzindo assim o risco de overfitting.

Os modelos foram treinados e avaliou-se as métricas de perda, acurácia, precisão e F1. Com base nos resultados obtidos, o modelo foi salvo e, em alguns casos, ajustou-se novamente os parâmetros de data augmentation e modificou-se a taxa de aprendizado, antes de continuar o treinamento por mais épocas.

A. Métricas

1) Acurácia: A acurácia é uma métrica que mede a proporção de predições corretas feitas pelo modelo em relação ao total de predições. É calculada pela Equação 1.

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Total de predições}} \tag{1}$$

2) Precisão: A precisão é a razão entre os verdadeiros positivos (tp) e a soma dos verdadeiros positivos e falsos positivos (fp). Ela representa a habilidade do classificador em não rotular como positiva uma amostra que é negativa. A precisão é calculada pela Equação 2.

$$\text{Precisão} = \frac{tp}{tp + fp} \tag{2}$$

3) Recall: O recall, também conhecido como revocação, é outra métrica importante em problemas de classificação. Essa métrica mede a proporção de exemplos positivos que foram corretamente identificados pelo modelo, sendo dada pela razão entre os verdadeiros positivos (tp) e a soma dos verdadeiros positivos e falsos negativos (fn). É calculado pela Equação 3.

$$\text{Recall} = \frac{tp}{tp + fn} \tag{3}$$

4) F1 Score: O F1 Score é a média harmônica entre precisão e recall. Ele fornece uma única métrica que equilibra precisão e recall. O intervalo de saída do F1 Score é de 0 a 1 e funciona tanto para classificação multi-classe quanto para classificação multi-rótulo. O F1 Score é calculado pela Equação 4.

$$\text{F1} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{4}$$

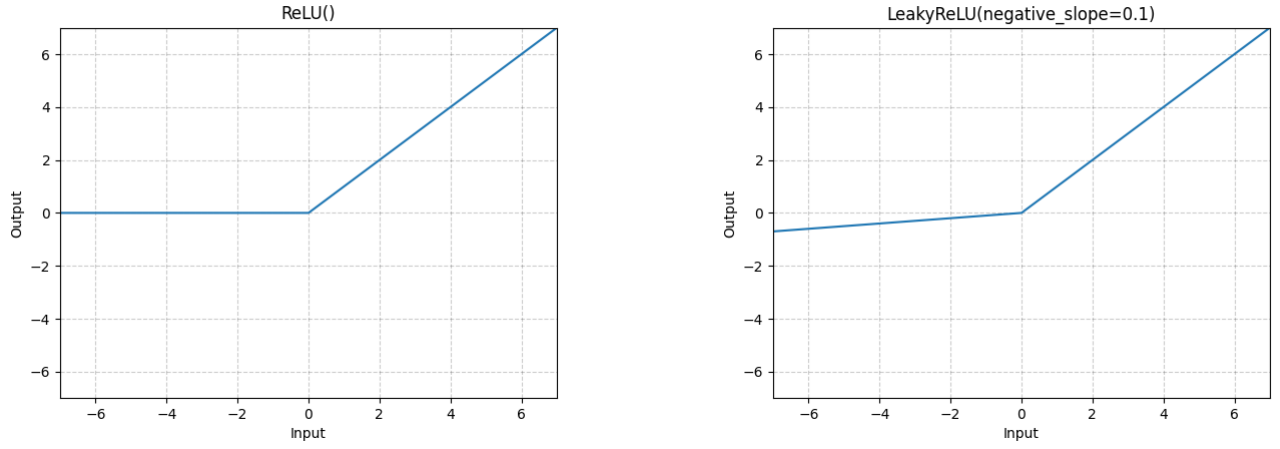
B. Funções de Ativação

As funções de ativação são componentes essenciais em redes neurais, pois introduzem não-linearidade nos modelos, permitindo que aprendam e representem relações complexas nos dados. Neste estudo, duas funções de ativação principais foram utilizadas: ReLU e Leaky ReLU. Outras funções de ativação, como tangente hiperbólica e sigmóide foram testadas em conjunto com o processo de escolha dos hiperparâmetros e não foram tão eficazes.

1) ReLU: A função de ativação ReLU é uma das mais populares e amplamente utilizadas em redes neurais profundas devido à sua simplicidade e eficiência. A ReLU ativa uma unidade passando os valores positivos diretamente e atribuindo zero aos valores negativos, o que ajuda a mitigar o problema do desaparecimento do gradiente em redes profundas. A função ReLU retorna o próprio valor se ele for positivo; caso contrário, retorna zero, conforme representado pela Equação 5 e visualizado na Figura 2a. Pode causar o problema de ”neurônios mortos”, onde algumas unidades podem parar de aprender se recebem sempre valores negativos.

$$ReLU(x) = max(0,\ x) \tag{5}$$

2) Leaky ReLU: A Leaky ReLU é uma variação da ReLU que resolve o problema dos ”neurônios mortos” ao permitir um pequeno gradiente para valores negativos. Em vez de retornar zero para valores negativos, a Leaky ReLU retorna uma fração desses valores. Esta função é representada pela Equação 6, em que m é um coeficiente negativo, e pode ser visualizada na Figura 2b.



(a) ReLU

(b) Leaky ReLU

Fig. 2: Funções de ativação utilizadas

$$Leaky\ Relu = \begin{cases} x & \text{se } x \geq 0 \\ m \cdot x & \text{se } x < 0 \end{cases} \quad (6)$$

C. Treinador

Para o treinamento dos modelos de redes neurais desenvolvidos neste estudo, foi utilizado o otimizador Gradiente Descendente Estocástico, do inglês Stochastic Gradient Descent (SGD). O SGD é um método de otimização que ajusta os pesos do modelo iterativamente com base no gradiente do erro em relação aos parâmetros do modelo. A fórmula utilizada para atualizar os parâmetros x no SGD é dada pela Equação 7, em que η é a taxa de aprendizado e $\nabla f_i(x)$ é o gradiente calculado a partir de um exemplo de treinamento selecionado aleatoriamente. Também foi testado o otimizador Adam. No entanto, os resultados obtidos com o Adam não foram satisfatórios.

$$x = x - \eta \nabla f_i(x) \quad (7)$$

Além do otimizador, foi empregado um scheduler de taxa de aprendizado, que ajusta a taxa de aprendizado de forma exponencial a cada época de treinamento, o que permite um controle mais refinado da taxa de aprendizado ao longo do tempo. A fórmula utilizada para atualizar a taxa de aprendizado η_t a cada época *t* é dada pela Equação 8, onde η_0 é a taxa de aprendizado inicial e γ é o fator de decaimento. Esse ajuste contínuo ajuda a evitar problemas de oscilação e estagnação no processo de treinamento.

$$\eta_t = \eta_0 \cdot \gamma^t \quad (8)$$

A utilização do ExponentialLR no treinamento dos modelos se mostrou eficaz, especialmente na fase de ajustes finos dos parâmetros, contribuindo para uma melhor generalização dos modelos aos dados de validação. Dessa forma, a combinação do otimizador SGD com o scheduler ExponentialLR proporcionou uma estratégia robusta para a otimização dos modelos de redes neurais desenvolvidos neste estudo.

D. Data Augmentation

As estratégias de data augmentation aplicadas neste estudo foram selecionadas para abordar uma ampla gama de transformações geométricas e fotométricas, com o objetivo de simular diferentes condições de visualização das imagens. Aqui está uma descrição das técnicas testadas:

- Random Horizontal Flip: Aplica uma probabilidade de girar horizontalmente as imagens. Valores de probabilidade aplicados até 50%.
- Random Vertical Flip: Aplica uma probabilidade de girar verticalmente as imagens. Valores de probabilidade aplicados até 30%.
- Random Affine: Aplica uma transformação afim aleatória, incluindo rotação, translação, escala e cisalhamento. Rotações aplicadas de até 20%. Escalas aplicadas entre 70% e 120%. Translações aplicadas de até 20% em ambos sentidos.
- Random Rotation: Rotaciona a imagem por um ângulo aleatório dentro de um intervalo especificado. Rotações aplicadas de até 70%.
- Random Resized Crop: Recorta e redimensiona a imagem de forma aleatória. Escalas aplicadas entre 70% e 120%.
- Color Jitter: Realiza ajustes aleatórios de brilho, contraste, saturação e matiz. Valores de brilho, contraste, saturação e matiz variados de 50% a 150%.
- Random Perspective: Aplica uma perspectiva aleatória na imagem. Valores para distorção da escala aplicados de até 30% com uma probabilidade de até 50%.
- Gaussian Blur: Aplica um desfoque gaussiano à imagem. Valores do desvio padrão utilizado para aplicar o desfoque entre 0,2 e 7.
- Random Erasing: Aplica um apagamento aleatório em uma região da imagem. Valores de probabilidade aplicados até 30%. Valores de escala aplicados de 5% a 40%. Valores para faixa de proporção da área aplicada entre 0,3 e 3,3.

E. Elementos que Compõem os Blocos de Construção dos Modelos

Os blocos de construção dos modelos de redes neurais convolucionais (CNN) implementados neste estudo utilizam vários elementos fundamentais, como camadas convolucionais e normalização de batches (Batch Normalization). Esses elementos são cruciais para a extração de características e para melhorar a eficácia e a eficiência dos modelos na tarefa de classificação de imagens. A seguir, detalhamos cada um desses componentes.

1) Camadas Convolucionais: As camadas convolucionais são a espinha dorsal das redes neurais convolucionais. A camada `Conv2d` realiza a operação de convolução em duas dimensões, que é essencial para detectar padrões locais nas imagens de entrada. A operação de convolução pode ser descrita matematicamente como na Equação 9, em que x é a entrada, w são os pesos do filtro e y é a saída da convolução.

$$y(i,\ j) = \sum_m \sum_n x(i+m,\ j+n) \cdot w(m,\ n) \quad (9)$$

As camadas convolucionais oferecem diversos benefícios importantes. Primeiramente, elas são altamente eficazes na detecção de padrões locais, como bordas, texturas e formas. Além disso, o uso de filtros compartilhados nas camadas convolucionais reduz o número total de parâmetros no modelo, o que diminui o risco de overfitting. Por fim, essas camadas ajudam a criar modelos que são invariantes a pequenas translações na imagem de entrada, o que melhora a robustez e a generalização do modelo.

2) Normalização de Batchs (Batch Normalization): A normalização de batches é uma técnica para normalizar as ativações de uma camada para cada mini-batch durante o treinamento. Isso é feito ajustando a média e a variância das ativações. A operação de normalização de batch pode ser descrita pelas seguintes Equações 10 e 11, em que μ_{batch} e σ^2_{batch} são a média e a variância calculadas ao longo do mini-batch, ϵ é um pequeno valor para evitar divisão por zero, e

γ e β são parâmetros aprendíveis que permitem à rede neural escalar e deslocar a normalização.

$$\hat{x} = \frac{x - \mu_{\text{batch}}}{\sqrt{\sigma^2_{\text{batch}} + \epsilon}} \tag{10}$$

$$y = \gamma\hat{x} + \beta \tag{11}$$

A normalização de batch oferece vários benefícios significativos. Primeiramente, ela permite o uso de taxas de aprendizado maiores, o que acelera o processo de treinamento. Além disso, ao normalizar as ativações, a normalização de batch reduz a sensibilidade do modelo à inicialização dos pesos. Por fim, essa técnica possui um efeito regularizador, o que pode diminuir a necessidade de utilizar outras técnicas de regularização, como o dropout.

A combinação dessas técnicas nos blocos de construção permite a criação de modelos de CNN mais robustos e eficientes, capazes de extrair características importantes das imagens de entrada e generalizar melhor para novos dados.

F. Blocos de Construção dos Modelos

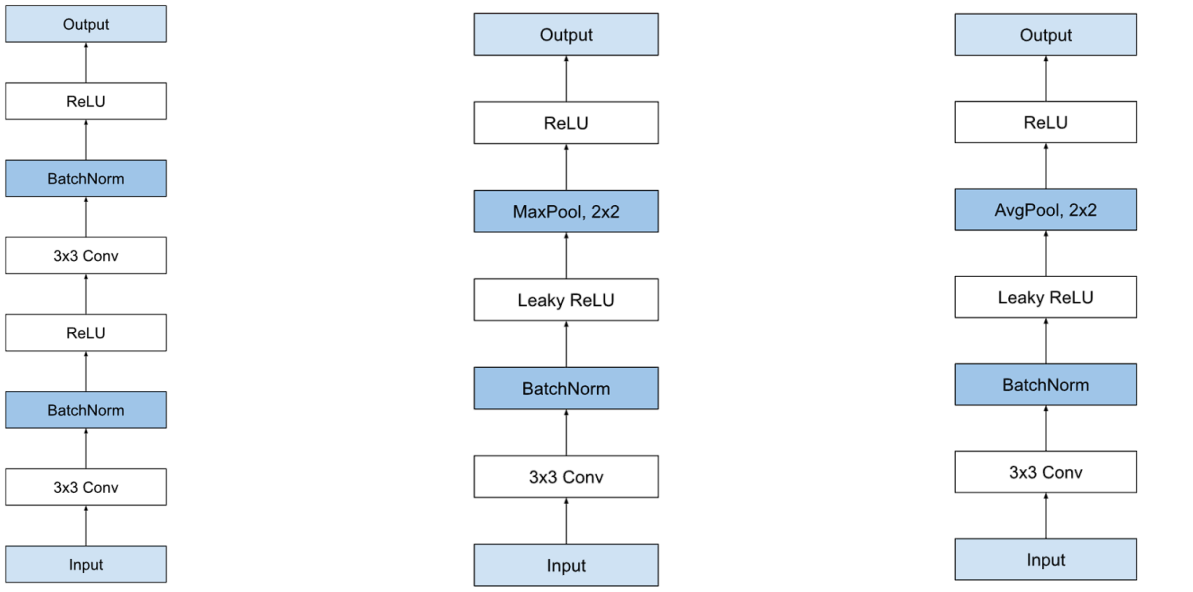
Os modelos de redes neurais convolucionais (CNN) implementados neste estudo são compostos por vários blocos de construção. Esses blocos são projetados para extrair características e realizar operações específicas nas entradas, contribuindo para a eficácia e eficiência dos modelos na tarefa de classificação de imagens. A seguir, é descrito cada um dos blocos testados.

1) Bloco Residual: O Bloco Residual é uma componente chave nas redes residuais (ResNet), uma arquitetura que revolucionou o campo das redes profundas. A principal inovação do Bloco Residual é a introdução de conexões residuais ou ”skip connections”. Essas conexões permitem que a entrada inicial do bloco seja somada diretamente à saída das camadas convolutivas, facilitando a propagação do gradiente e mitigando o problema de desaparecimento do gradiente em redes muito profundas. A estrutura deste bloco, conforme ilustrado na Figura 3a, consiste em duas camadas convolutivas seguidas de uma normalização em batch (BatchNorm). A ativação ReLU é aplicada após cada convolução e normalização. Este design permite que a rede aprenda funções de mapeamento de identidade, tornando mais fácil a aprendizagem de redes mais profundas.

2) Bloco de Pooling Máximo: O Bloco de Pooling Máximo é fundamental para a redução da dimensionalidade espacial das características extraídas, preservando ao mesmo tempo as informações mais salientes. Este bloco é composto por uma camada convolutiva, seguida por uma normalização em batch e uma operação de pooling máximo. A convolução atua na extração de características locais, enquanto o pooling máximo reduz a resolução espacial da entrada ao selecionar o valor máximo em cada região de pooling, ajudando a aumentar a robustez da rede às variações e distorções nas imagens de entrada. A estrutura deste bloco pode ser observada na Figura 3b.

3) Bloco de Pooling Médio: O Bloco de Pooling Médio é semelhante ao Bloco de Pooling Máximo, com a diferença de que utiliza a operação de pooling médio em vez do pooling máximo. O pooling médio calcula a média dos valores em cada região de pooling, em vez de selecionar o valor máximo. Essa abordagem pode ser útil para capturar uma representação mais suave das características extraídas, reduzindo a sensibilidade a ruídos e variações extremas nas entradas. A estrutura deste bloco também é ilustrada na Figura 3c.

4) Bloco Inception: O Bloco Inception é inspirado na arquitetura GoogleLeNet (Inception), conhecida por sua capacidade de capturar características em múltiplas escalas através



(a) Bloco Residual (b) Bloco de Pooling Máximo (c) Bloco de Pooling Médio

Fig. 3: Blocos de Construção dos Modelos

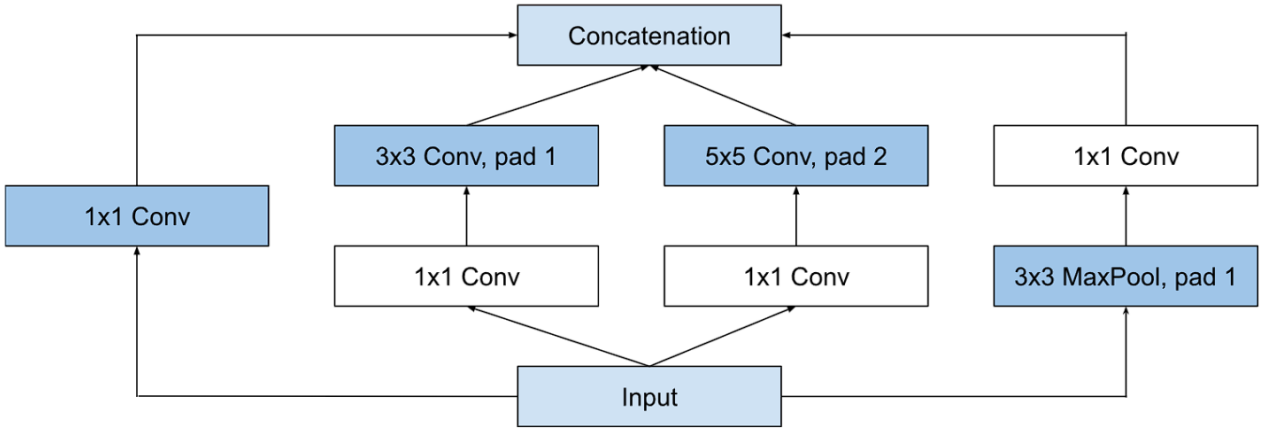


Fig. 4: Bloco Inception

de filtros de diferentes tamanhos operando em paralelo. Este bloco é composto por várias ramificações (branches), cada uma aplicando convoluções com diferentes tamanhos de filtro (1x1, 3x3, 5x5) e uma operação de pooling. As saídas de todas as ramificações são concatenadas ao longo da dimensão dos canais, combinando as informações extraídas de diferentes escalas. Essa abordagem permite que a rede aprenda representações ricas e diversificadas das entradas. A Figura 4 ilustra a estrutura detalhada do Bloco Inception.

G. Modelo 1

O desenvolvimento do Modelo 1 foi guiado pela necessidade de equilibrar complexidade e desempenho. Este modelo é composto por diversos blocos primários de construção, tais como blocos de pooling máximo e médio, além dos blocos residuais e de pooling máximo e médio.

O modelo começa com um bloco de pooling máximo (*MaxPoolBlock*), seguido por um bloco residual (*ResidualBlock*). A ideia inicial foi testar a eficácia dos blocos residuais inspirados em redes ResNet, que são conhecidos por mitigar o problema de desaparecimento do gradiente em redes profundas, facilitando a aprendizagem. Em seguida, adicionou-se outro bloco de pooling máximo e um bloco residual para aumentar a profundidade do modelo.

Posteriormente, foi incorporado um bloco de pooling médio (*AvgPoolBlock*) para complementar os blocos de pooling máximo, visando avaliar se a combinação das duas técnicas de pooling poderia melhorar a extração de características. A estrutura final do modelo inclui uma camada densa composta por operações de pooling máximo, achatamento dos dados (*flatten*), e múltiplas camadas totalmente conectadas (*fully connected*), intercaladas com camadas de dropout para evitar overfitting.

Inicialmente, testou-se diversas combinações de blocos para encontrar a arquitetura mais eficaz. Isso incluiu:

- Um modelo com blocos residuais (inspirado em ResNet).
- Um modelo com um maior número de camadas para verificar o impacto da profundidade na performance.
- Um modelo utilizando apenas blocos de pooling médio (*AvgPool*) para entender a eficácia dessa técnica de pooling.
- Um modelo utilizando apenas blocos de pooling máximo (*MaxPool*) para comparar com o modelo misto.

Após uma série de experimentos, constatou-se que a remoção dos blocos residuais resultou em uma melhoria na acurácia do modelo. A análise revelou que, para o nosso conjunto de dados específico, a inclusão dos blocos residuais não oferecia benefícios significativos e, em alguns casos, complicava o processo de treinamento.

A Tabela I apresenta um resumo detalhado da arquitetura do Modelo 1, destacando a configuração final utilizada após a otimização dos testes mencionados. A escolha de cada componente foi baseada em resultados empíricos obtidos durante a fase de experimentação.

TABLE I: Resumo da Arquitetura do Modelo 1

Camada	Descrição
MaxPoolBlock	Entrada: 3 canais, Saída: 32 canais
ResidualBlock	Entrada/Saída: 32 canais
MaxPoolBlock	Entrada: 32 canais, Saída: 64 canais
ResidualBlock	Entrada/Saída: 64 canais
MaxPoolBlock	Entrada: 64 canais, Saída: 128 canais
AvgPoolBlock	Entrada/Saída: 128 canais
Dense	MaxPool, Flatten, Fully Connected, Dropout

O modelo 1 final possui 904.420 parâmetros totais, sendo todos treináveis. E os hiperparâmetros utilizados no seu treino foram:

- taxa de aprendizado (lr): variável entre 10^{-3} e $8 \cdot 10^{-6}$
- decaimento de pesos (weight decay): fixo de 10^{-3}
- dropout: fixo de 0,3 ou 30%
- fator multiplicativo da queda da taxa de aprendizado: fixo de 0,95

H. Modelo 2

O Modelo 2 foi desenvolvido com o objetivo de explorar a eficiência dos blocos de Inception na extração de

características de imagens. A arquitetura foi inspirada na GoogleNet (Inception), conhecida por sua capacidade de capturar informações em múltiplas escalas simultaneamente. A estrutura do Modelo 2 combina blocos de Inception com camadas convolucionais e técnicas de pooling, resultando em uma arquitetura robusta e eficaz para a tarefa de classificação de imagens.

O modelo começa com uma camada convolucional (*Conv2d*) seguida por uma operação de pooling máximo (*MaxPool2d*). Esta etapa inicial visa reduzir a dimensionalidade da entrada e extrair características básicas. Em seguida, outra camada convolucional é aplicada, seguida por outra operação de pooling máximo, para refinar ainda mais as características extraídas.

A parte central do modelo é composta por dois blocos de Inception. Cada bloco de Inception aplica múltiplos filtros convolucionais de diferentes tamanhos em paralelo, capturando características em várias escalas. O primeiro bloco de Inception recebe 192 canais de entrada e gera uma combinação de saídas com diferentes dimensões. O segundo bloco de Inception expande essa combinação, recebendo 256 canais de entrada e processando-os para capturar características mais complexas.

Após os blocos de Inception, uma terceira operação de pooling máximo é aplicada para reduzir ainda mais a dimensionalidade e agregar as informações extraídas. Um bloco de pooling médio (*AvgPool2d*) é então utilizado, seguido por uma camada de dropout para evitar overfitting. A fase final do modelo é composta por uma camada totalmente conectada (*fully connected*), que mapeia as características extraídas para as classes de saída.

Para validar a arquitetura, várias combinações de blocos foram testadas. Isso incluiu:

- Um modelo com apenas um bloco de Inception.
- Um modelo com mais blocos de Inception para avaliar o impacto de uma maior profundidade.
- Um modelo com mais camadas convolucionais.
- Um modelo utilizando somente pooling médio (*AvgPool*) para comparar com o uso combinado de pooling máximo e médio.

A Tabela II apresenta um resumo detalhado da arquitetura do Modelo 2, destacando a configuração final utilizada após a otimização dos testes mencionados. A escolha de cada componente foi baseada em resultados empíricos obtidos durante a fase de experimentação.

O modelo 2 final possui 3.024.788 parâmetros totais, sendo todos treináveis. E os hiperparâmetros utilizados no seu treino foram:

- taxa de aprendizado (lr): variável entre 10^{-3} e $6 \cdot 10^{-5}$
- decaimento de pesos (weight decay): fixo de 10^{-2}
- dropout: fixo de 0,35 ou 35%
- fator multiplicativo da queda da taxa de aprendizado: fixo de 0,95

III. Simulações e Resultados

A. Modelo 1

O modelo 1 foi treinado utilizando uma arquitetura baseada em blocos de convolução e pooling durante 1210 épocas. Nos gráficos da Figura 5 e na Tabela III são apresentados os resultados obtidos durante o treinamento e teste deste modelo.

TABLE II: Resumo da Arquitetura do Modelo 2

Camada	Descrição
Conv2d	Entrada: 3 canais, Saída: 64 canais, Kernel: 7x7, Stride: 2, Padding: 3
MaxPool2d	Kernel: 3x3, Stride: 2, Padding: 1
Conv2d	Entrada: 64 canais, Saída: 192 canais, Kernel: 3x3, Stride: 1, Padding: 1
MaxPool2d	Kernel: 3x3, Stride: 2, Padding: 1
InceptionBlock1	Entrada: 192 canais, Saída: combinação de várias dimensões
InceptionBlock2	Entrada: 256 canais, Saída: combinação de várias dimensões
MaxPool2d	Kernel: 3x3, Stride: 2, Padding: 1
AvgPool2d	Kernel: 2x2
Dropout	Probabilidade: variável
Fully Connected	Entrada: 23520 características, Saída: 100 classes

TABLE III: Resultados do Modelo 1

Época	Perda no treino	Acurácia no treino	Perda no teste	Acurácia no teste
107	2,18	43,31%	1,86	50,80%
324	1,16	68,94%	1,33	63,60%
335	1,08	71,21%	1,37	61,60%
619	0,8	77,88%	1,01	70,80%
704	0,93	73,88%	1,15	69,00%
851	0,74	80,26%	0,85	75,20%
1043	0,51	88,55%	0,66	81,60%
1049	0,5	88,42%	0,65	82,00%

Para avaliar o desempenho do Modelo 1, realizou-se a validação utilizando o conjunto de dados separado especificamente para essa finalidade. O modelo obteve uma acurácia geral de 92,20%. No entanto, ao analisar individualmente cada classe, observou-se uma distribuição variada de acurácias, conforme detalhado a seguir:

- Acurácia de 0%: 0 classes
- Acurácia de 20%: 0 classes
- Acurácia de 40%: 1 classe
- Acurácia de 60%: 5 classes
- Acurácia de 80%: 26 classes
- Acurácia de 100%: 68 classes

A Figura 6 ilustra exemplos de imagens com as respectivas classes reais e as classes preditas pelo Modelo 1.

B. Modelo 2

O modelo 2 foi treinado utilizando uma arquitetura baseada em blocos de convolução, pooling e bloco Inception durante 490 épocas. Nos gráficos da Figura 7 e na Tabela IV são apresentados os resultados obtidos durante o treinamento e teste deste modelo.

Para avaliar o desempenho do Modelo 2, realizou-se a validação utilizando o conjunto de dados separado especificamente para essa finalidade. O modelo obteve uma acurácia geral de 98,40%. No entanto, ao analisar individualmente

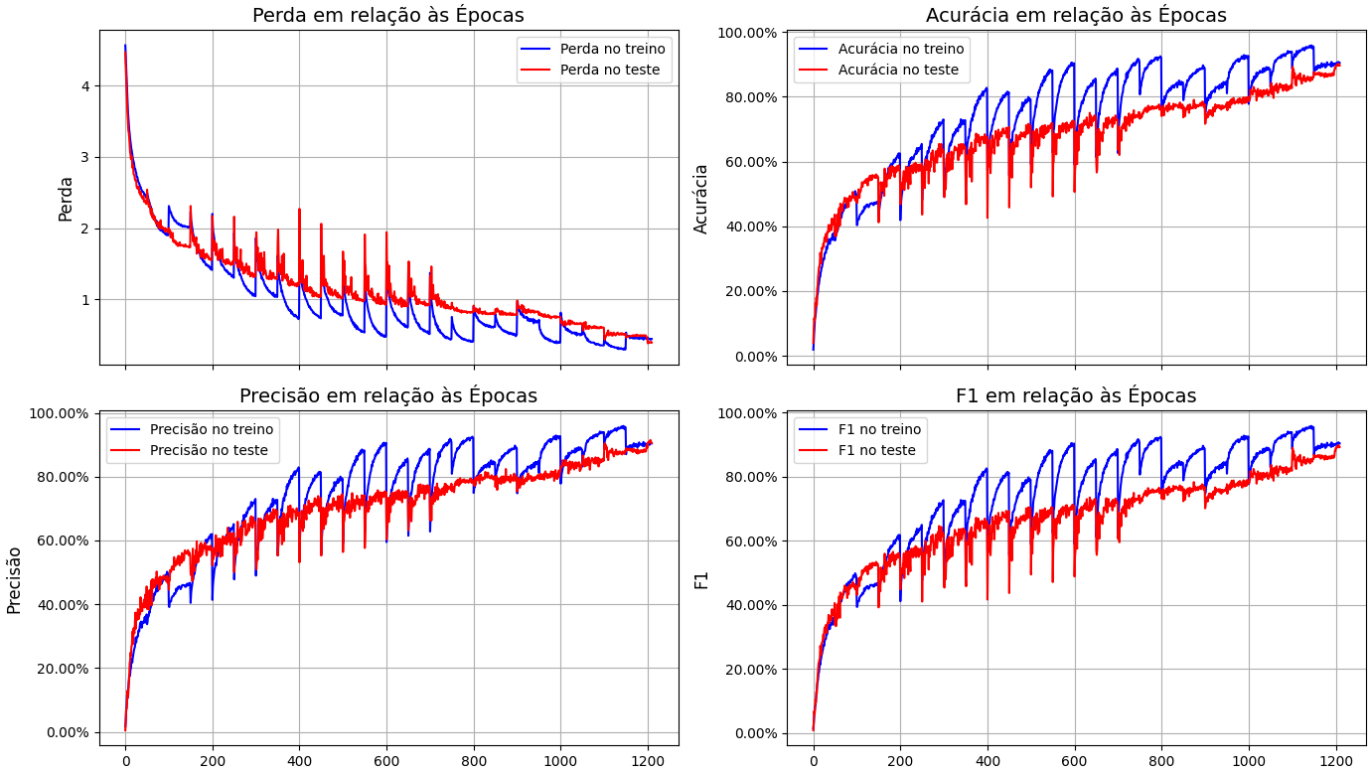


Fig. 5: Resultados do Modelo 1

Fig. 6: Validação do Modelo 1

cada classe, observou-se uma distribuição variada de acurácias, conforme detalhado a seguir:

- Acurácia de 0%: 0 classes
- Acurácia de 20%: 0 classes
- Acurácia de 40%: 0 classes
- Acurácia de 60%: 0 classes
- Acurácia de 80%: 8 classes
- Acurácia de 100%: 92 classes

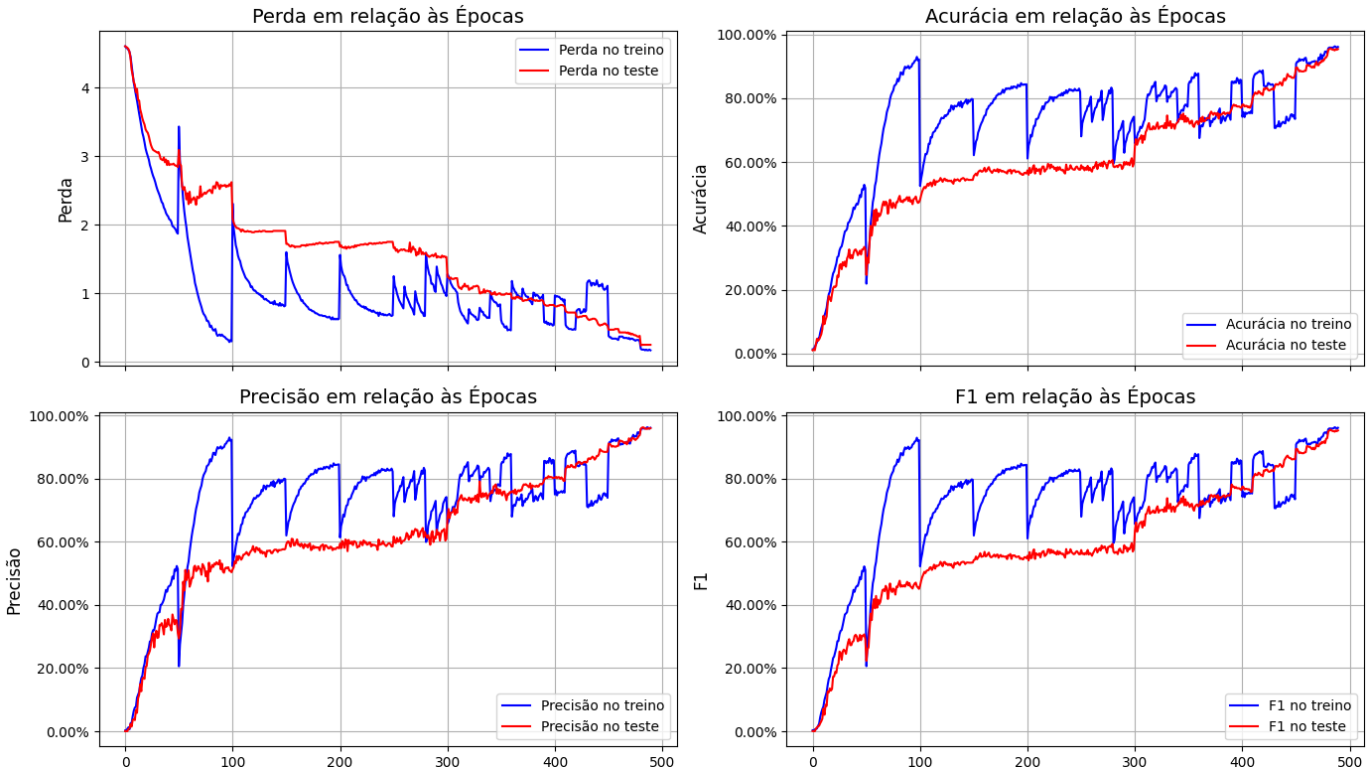


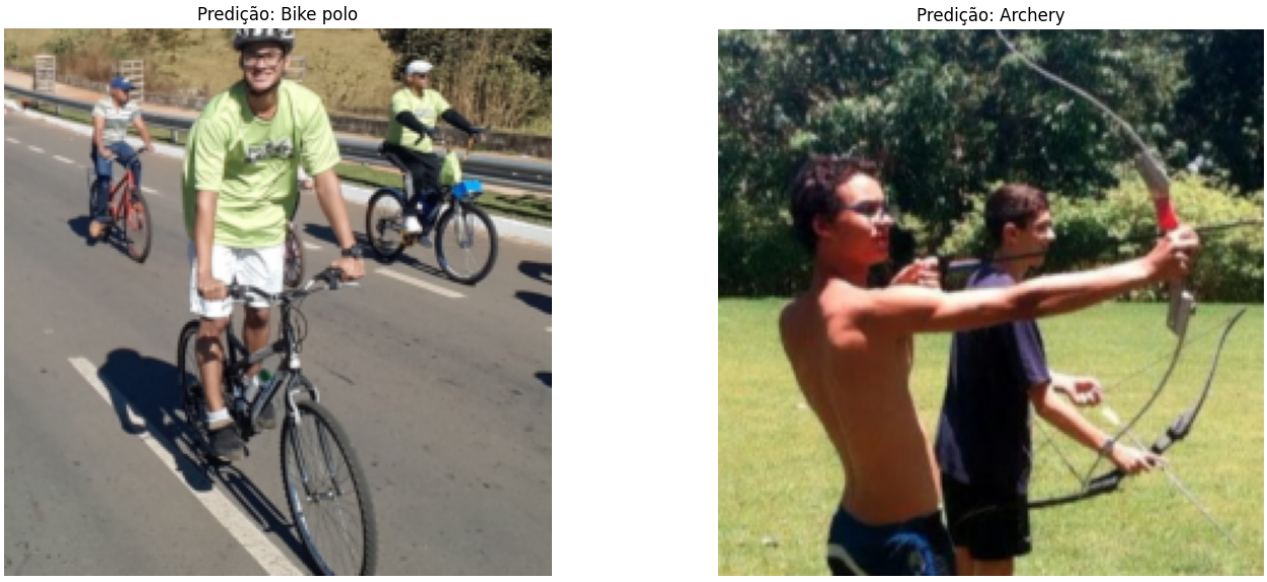
Fig. 7: Resultados do Modelo 2

Fig. 8: Validação do Modelo 2

A Figura 8 ilustra exemplos de imagens com as respectivas classes reais e as classes preditas pelo Modelo 2.

C. *Teste com Imagens Próprias*

O desempenho dos modelos foi avaliado utilizando imagens fornecidas pelo autor, o resultado das previsões foram registradas na Figura 9.



(a) Modelo 1 (b) Modelo 2

Fig. 9: Aplicação dos Modelos para Predicção em Imagens Próprias

TABLE IV: Resultados do Modelo 2

Época	Perda no treino	Acurácia no treino	Perda no teste	Acurácia no teste
27	2,67	33,58%	3,08	26,40%
33	2,38	40,97%	2,96	30,60%
253	1,0	74,50%	1,64	57,40%
388	0,94	74,30%	0,9	75,20%
409	0,91	75,27%	0,81	78,80%
419	0,47	88,73%	0,72	81,00%
455	0,34	92,04%	0,47	88,40%
464	0,35	91,22%	0,43	89,80%

IV. Conclusão

Os resultados deste estudo destacam a aplicação de modelos de redes neurais convolucionais na tarefa de identificação de diferentes esportes a partir de imagens. Através da aplicação de técnicas convolucionais, regularização e data augmentation, os Modelos 1 e 2 alcançaram uma acurácia nos conjuntos de validação de 92,20% e 98,40%, respectivamente.

A utilização de técnicas de data augmentation foi fundamental para aumentar a robustez dos modelos, permitindo-lhes generalizar melhor para novos dados. Além disso, os testes mostraram que nem sempre um modelo mais complexo é mais eficiente, ao remover os blocos residuais do modelo 1.

Felipe Fonseca é desenvolvedor fullstack e apaixonado por tecnologia. Atualmente cursa Engenharia Química na Universidade Federal de Goiás, onde participou de diversos projetos e desafios que ampliaram seu conhecimento e habilidades interpessoais. Felipe também está cursando duas disciplinas como aluno especial do mestrado em Engenharia de Computação na UFG.