UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE VETERINÁRIA E ZOOTECNIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ZOOTECNIA

**CLASSIFICAÇÃO DE RAÇAS DE BOVINOS EM SUBESPÉCIES**

**(*Bos taurus taurus* e *Bos taurus indicus*)**

**Disciplina**: Inteligência artificial aplicada à Ciência Animal

**Docente**: Prof. Dr. Sergio Silva

**Discentes**: Ana Caroline Rodrigues da Cunha

Giovanna Vargas Barbosa

Guilherme Brandão Gonçalves Bizinoto

Vicente Batista de Souza Junior

GOIÂNIA - GO

2024

1. **INTRODUÇÃO**

A pecuária bovina atualmente é uma das principais atividades agropecuárias do mundo, e a mesma desempenha um grande papel na segurança alimentar da população mundial, tanto se tratando da disponibilidade de proteína de origem animal e produção de leite. O Brasil atualmente assume a posição de segundo maior produtor e maior exportador de carne bovina do mundo (ABIEC, 2023).

Os bovinos são hoje representados por cerca de 800 raças, numa população mundial de 1,4 bilhões de animais, sendo identificadas em subespécies taurinos (*bos taurus*) e zebuínos (*bos indicus*), sendo que, as subespécies apresentam padrões genéticos , características produtivas, morfológicas e fenotípicas diferentes entre si, o que por sua vez refletem na adaptabilidade de cada subespécie a cada tipo de sistema de produção bem como a cada região e clima que os animais podem ser produzidos.

Na prática, a variabilidade genética influencia nos índices produtivos de leite, qualidade do leite, índices produtivos de carcaça, qualidade de carcaça, na rusticidade e resistência a doenças, e climas mais quentes e áridos, além da total relação da subespécie com o desempenho dos animais (consumo de alimento, eficiência alimentar, ganho de peso etc…), essas informações e características têm grande influência principalmente em programas de cruzamentos e melhoramento genéticos de bovinos(Mendonça., et al. 2019; Lima Júnior et al., 2011; Neto., et al. 2014).

A classificação de raças de bovinos a partir de fotos, via redes neurais convolucionais, oferece eficiência no trabalho e melhoria na gestão da pecuária. A inteligência artificial (IA), especialmente os modelos de aprendizado de máquina, podem ser treinados para reconhecer padrões visuais específicos de diferentes raças de bovinos. Uma vez treinado, o modelo pode classificar as raças com alta precisão e consistência, reduzindo o erro humano. Isso é especialmente importante em ambientes com grandes volumes de dados, onde a classificação manual seria demorada e propensa a erros. Em grandes fazendas ou em confinamentos, onde há centenas ou milhares de bovinos, a IA permite escalar a identificação e monitoramento das raças de forma eficiente.

Embora existam diversas metodologias que vêm sendo empregadas na produção pecuária, muitas delas são de baixa eficiência e/ou necessitam de mão de obra, tempo ou recursos financeiros, ao mesmo tempo disso, a aplicação de machine learning para essa finalidade é pouco utilizada no Brasil.

Neste estudo, desenvolvemos e avaliamos a eficiência de um modelo de aprendizagem de máquina com o enfoque na classificação de bovinos entre as subespécies (zebuínos e taurinos) a partir de imagens, com o objetivo de demonstrar a viabilidade da ferramenta nos sistemas de produção de bovinos.

1. **MATERIAL E MÉTODOS**

Foi utilizado o banco de dados Roboflow .e Kaggle, com 1.900 imagens constando subespécies de bovinos de corte, as quais foram divididas em 2 classes (taurinos e zebuínos). Além disso, outras bases foram consultadas, como: Pexels, Freepik e Github, visando obter o número de imagens suficientes para alimentar a rede, com objetivo de aumentar a acurácia.

Para a realização do trabalho foi utilizada as redes neurais convolucionais (CNN). As CNNs são uma classe de redes neurais profundas (deep learning) projetadas especificamente para o processamento de dados com uma estrutura de grade, como imagens. Elas são amplamente utilizadas em tarefas de visão computacional, como classificação de imagens, detecção de objetos, segmentação de imagens, e até mesmo em áreas não relacionadas a imagens, como processamento de sinais e texto.

O principal objetivo das CNNs é extrair características hierárquicas dos dados de entrada, usando uma combinação de camadas convolucionais, camadas de pooling e camadas totalmente conectadas. A arquitetura é projetada para capturar padrões locais, invariâncias espaciais e aprender representações robustas de dados.

A arquitetura utilizada no trabalho foi a VGG16. A VGG16 é uma das arquiteturas mais famosas de redes neurais convolucionais (CNNs), criada pelo Visual Geometry Group (VGG) e é composta por 16 camadas de aprendizado, sendo 13 camadas convolucionais e 3 camadas totalmente conectadas. A principal característica do modelo VGG16 é a simplicidade em sua construção, com a utilização de filtros convolucionais pequenos (3×3) e pooling com janela 2×2, o que ajuda a rede a ter profundidade sem um aumento excessivo na complexidade computacional (Simonyan e Zisserman, 2014)

**2.1 Análise e processamento das imagens**

As imagens foram processadas com o auxílio da ferramenta *Google Colaboratory*, com a aplicação de pacotes na linguagem Python, utilizando as bibliotecas *os, sklearn, matplolib, tensorflow, keras, numpy e seaborn*. De 1900 imagens de entrada, subdividiu-se em 70% para o treinamento, 15% para validação e 15% para teste para as duas classes (subespécies) e as imagens foram particionadas em seus respectivos diretórios.

**import os**

**import shutil**

**from sklearn.model\_selection import train\_test\_split**

**# Caminhos principais**

**base\_path\_zebuinos = '/content/drive/MyDrive/IA\_ciencia\_animal/TrabalhoIA/'**

**source\_dir = os.path.join(base\_path\_zebuinos, 'Cattle/Zebuinos')**

**target\_base\_dir = os.path.join(base\_path\_zebuinos, 'Images/Zebuinos')**

**# Proporções para train, validation e test**

**train\_ratio = 0.7**

**validation\_ratio = 0.15**

**test\_ratio = 0.15**

**# Diretórios de destino**

**train\_dir = os.path.join(target\_base\_dir, 'train')**

**validation\_dir = os.path.join(target\_base\_dir, 'validation')**

**test\_dir = os.path.join(target\_base\_dir, 'test')**

**# Criar diretórios de destino, se não existirem**

**for directory in [train\_dir, validation\_dir, test\_dir]:**

**os.makedirs(directory, exist\_ok=True)**

**# Listar todas as imagens no diretório de origem**

**images = [img for img in os.listdir(source\_dir) if img.lower().endswith(('png', 'jpg', 'jpeg'))]**

**# Garantir que há imagens suficientes**

**if len(images) < 3:**

**raise ValueError("O diretório de origem não contém imagens suficientes para dividir.")**

**# Divisão em treino, validação e teste**

**train\_files, temp\_files = train\_test\_split(images, train\_size=train\_ratio, random\_state=42)**

**validation\_files, test\_files = train\_test\_split(temp\_files, test\_size=test\_ratio / (validation\_ratio + test\_ratio), random\_state=42)**

**# Função para mover arquivos para o diretório apropriado**

**def move\_files(file\_list, destination\_dir):**

**for file\_name in file\_list:**

**source\_path = os.path.join(source\_dir, file\_name)**

**target\_path = os.path.join(destination\_dir, file\_name)**

**shutil.move(source\_path, target\_path)**

**print(f"Arquivo '{file\_name}' movido para '{destination\_dir}'")**

**# Mover arquivos para os respectivos diretórios**

**move\_files(train\_files, train\_dir)**

**move\_files(validation\_files, validation\_dir)**

**move\_files(test\_files, test\_dir)**

**print("Particionamento concluído.")**

**print("Treino:", len(train\_files))**

**print("Validação:", len(validation\_files))**

**print("Teste:", len(test\_files))**

Posteriormente, as bibliotecas foram importadas e definida a utilização do modelo CNN utilizando a VGG16 pelo tensorflow.keras.applications.

import sys

from matplotlib import pyplot as plt

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten, Dropout

from tensorflow.keras.optimizers import SGD, Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

import os

# NO CASO ESTÁ IMPORTANDO A CNN VGG16

from tensorflow.keras.applications import VGG16

Utilizou-se uma camada totalmente condensada (Dense) com 128 neurônios e função de ativação ReLu nas camadas ocultas, bem como adicionada uma camada do tipo "dropout" com 20% dos neurônios descartados aleatoriamente durante cada passagem da convolução para evitar overfitting. Ainda, foi utilizado o otimizador 'Adam' para ajustar as taxas de aprendizado de 0.0001 e a função de perda para classificação com mais de uma classe (taurinos e zebuínos). As métricas utilizadas para avaliar os resultados do modelo de classificação foram a 'acurácia', 'precisão', 'revocação' predefinidas no Keras e, 'escore F1'.

# define cnn model using VGG16

def define\_model(nclasses, input\_shape):

# PARA TESTAR OUTRAS CNNS, ALÉM DA ALTERAÇÃO DA IMPORTAÇÃO DESSA ACIMA,

# VC VAI TROCAR VGG16 PELO NOME DA CNN IMPORTADA

# Load the VGG16 model, excluding the top (fully connected) layers

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=input\_shape)

# Freeze the layers in the base model

for layer in base\_model.layers:

layer.trainable = False

# Create a new model

model = Sequential()

# Add the VGG16 base model

model.add(base\_model)

# Flatten the output from the base model

model.add(Flatten())

# Add new fully connected layers

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.2))

# Output layer with num\_classes units and softmax activation

model.add(Dense(2, activation='softmax'))

# Compile the model with categorical crossentropy for multi-class classification

opt = Adam(learning\_rate=0.0001)

model.compile(optimizer=opt, loss='categorical\_crossentropy', metrics=['accuracy', 'precision', 'recall', 'F1Score'])

return model

# plot diagnostic learning curves

def summarize\_diagnostics(history):

# plot loss

plt.subplot(211)

plt.title('Cross Entropy Loss')

plt.plot(history.history['loss'], color='blue', label='train')

plt.plot(history.history['val\_loss'], color='orange', label='test')

# plot accuracy

plt.subplot(212)

plt.title('Classification Accuracy')

plt.plot(history.history['accuracy'], color='blue', label='train')

plt.plot(history.history['val\_accuracy'], color='orange', label='test')

# save plot to file

filename = sys.argv[0].split('/')[-1]

plt.savefig(filename + '\_plot.png')

plt.close()

# Funcao principal

def funcao\_principal(num\_classes, base\_path, input\_shape, model\_name, epochs):

# chama a função que define o modelo da rede

model = define\_model(num\_classes, input\_shape)

# cria o gerador de dados de treino fazendo a normalização dos valores de pixels na escala real [0,1] e tirando 30% das amostras para validacao

datagen\_train = ImageDataGenerator(rescale=1.0/255.0, validation\_split=0.15)

# cria o gerador de dados de teste fazendo a normalização dos valores de pixels na escala real [0,1]

datagen\_test = ImageDataGenerator(rescale=1.0/255.0)

# prepara os iteradores

batch\_size = 32

train\_it = datagen\_train.flow\_from\_directory(os.path.join(base\_path, 'train'),

class\_mode='categorical', batch\_size=batch\_size, target\_size=(200, 200))

validation\_it = datagen\_train.flow\_from\_directory(os.path.join(base\_path, 'validation'),

class\_mode='categorical', batch\_size=batch\_size, target\_size=(200, 200))

test\_it = datagen\_test.flow\_from\_directory(os.path.join(base\_path,'test'),

class\_mode='categorical', target\_size=(200, 200))

# Define callback/heurística para salvar o melhor modelo com base em val\_loss

checkpoint = ModelCheckpoint(filepath=os.path.join(base\_path, model\_name), monitor='val\_loss', save\_best\_only=True, mode='min', verbose=1)

# Define callback/heurística de redução de taxa de aprendizado sob plateau

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=3, min\_lr=1e-5, verbose=1)

# Define callback/heurística para parar o treinamento mais cedo em caso de estagnação. Evita overfitting.

early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, verbose=1, restore\_best\_weights=True)

# fit model

history = model.fit(train\_it, validation\_data=validation\_it,

validation\_steps=validation\_it.samples//batch\_size, epochs=epochs,

callbacks=[checkpoint, reduce\_lr, early\_stopping],verbose=2)

Criou-se uma instância de *ImageDataGenerator* para aumentar e pré-processar imagens de treinamento fazendo a normalização dos valores de pixels na escala real [0,1] e tirando 15% das amostras para validação, bem como também para os dados de teste fazendo a normalização dos valores de pixels na escala real [0,1] em 15% das imagens, objetivando pré-processamento e aumento dos dados em tempo real de lotes. Foi redimensionado os valores de pixel para ficarem entre 0 e 1 (do intervalo original de 0 a 255) e a normalização de valores de pixel é uma etapa comum de pré-processamento para convergência mais rápida durante o treinamento.

Foi solicitada a geração de 30 epochs para acionar o treinamento do modelo com base em nclasses = 2.

input\_shape = (200, 200, 3)

epochs = 30

base\_path = '/content/drive/MyDrive/IA\_ciencia\_animal/TrabalhoIA/Images'

model\_name = 'vgg16\_cattle.keras'

nclasses = 2

funcao\_principal(nclasses, base\_path, input\_shape, model\_name, epochs)

A avaliação do modelo foi realizada pelas métricas acima citadas e obteve-se as previsões do modelo determinando uma matriz de confusão entre a classe verdadeira e a classe predita, isto é, o índice da classe com maior probabilidade de acertos.

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

import seaborn as sns

base\_path = '/content/drive/MyDrive/IA\_ciencia\_animal/TrabalhoIA/Images'

model\_name = 'vgg16\_cattle.keras'

# cria o gerador de dados de teste fazendo a normalização dos valores de pixels na escala real [0,1]

datagen\_test = keras.preprocessing.image.ImageDataGenerator(rescale=1.0 / 255.0)

test\_it = datagen\_test.flow\_from\_directory(

os.path.join(base\_path, 'test'),

class\_mode='categorical',

target\_size=(200, 200),

shuffle=False # Importante para manter as classes alinhadas

)

model = keras.models.load\_model(os.path.join(base\_path, model\_name))

# Avalia o modelo

loss, accuracy, precision, recall, f1\_score = model.evaluate(test\_it, verbose=2)

print(f"Perda: {loss:.3f}")

print(f"Acurácia: {accuracy \* 100.0:.3f}%")

print(f"Precisão: {precision \* 100.0:.3f}")

print(f"Revocação: {recall \* 100.0:.3f}")

# Obter as previsões do modelo

predictions = model.predict(test\_it, verbose=2)

# A classe verdadeira de cada imagem

true\_classes = test\_it.classes

# Obter as previsões como uma classe, ou seja, o índice da classe com a maior probabilidade

predicted\_classes = np.argmax(predictions, axis=1)

# Calcular a matriz de confusão com TensorFlow

conf\_matrix\_tf = tf.math.confusion\_matrix(

labels=true\_classes,

predictions=predicted\_classes

).numpy()

print(conf\_matrix\_tf)

# Mostrar a matriz de confusão com o seaborn

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix\_tf, annot=True, fmt='d', cmap='Blues',

xticklabels=test\_it.class\_indices.keys(),

yticklabels=test\_it.class\_indices.keys())

plt.title('Matriz de Confusão (TensorFlow)')

plt.xlabel('Classe Predita')

plt.ylabel('Classe Verdadeira')

plt.show()

Segue o link para acesso ao código aberto pelo Google Colab: <https://colab.research.google.com/drive/1o8BenuS3erfdk6Ay3KD0ZsvT6iJ9HS-8?authuser=2#scrollTo=PXigYKLLqOQG>.

1. **RESULTADOS E DISCUSSÃO**

Os resultados do modelo de classificação entre as subespécies de bovinos, zebuínos e taurinos, mostram um desempenho geral robusto. Durante o treinamento, o modelo atingiu uma precisão de 99,33%, com um F1-Score de 0,9933 e um loss de 0,0584, indicando alta eficiência em se ajustar aos dados de treino. Na validação, houve uma leve queda de desempenho, com precisão de 92,01%, F1-Score de 0,9197 e loss de 0,1739, sugerindo um pequeno indício de overfitting, mas ainda mantendo boa capacidade de generalização.

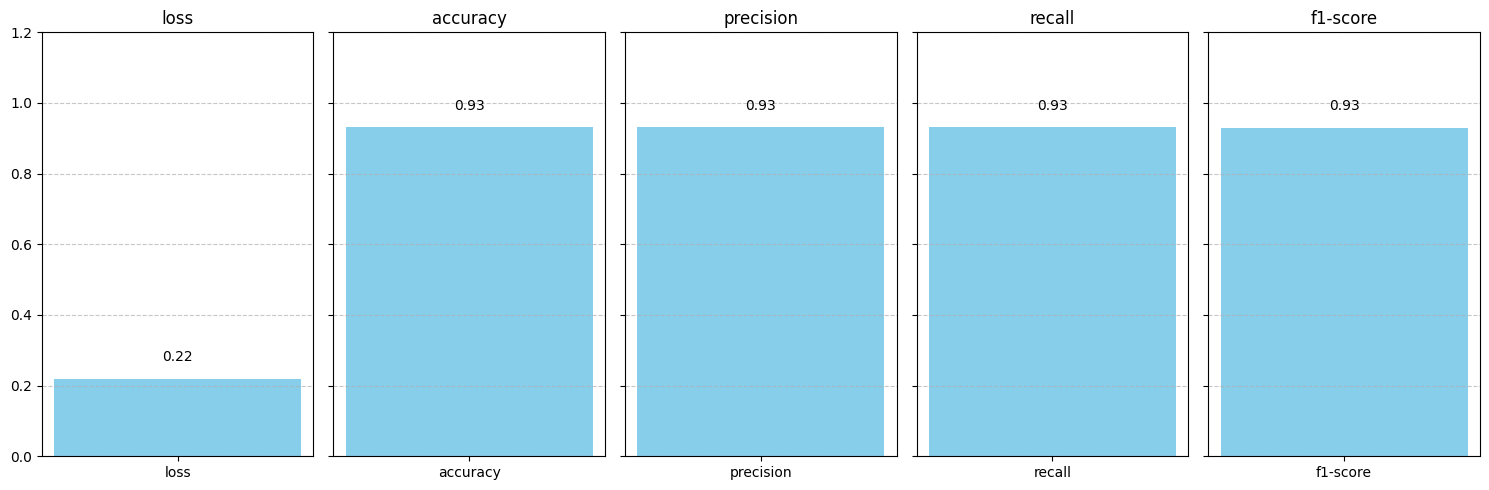


Imagem 1: Métricas de avaliação

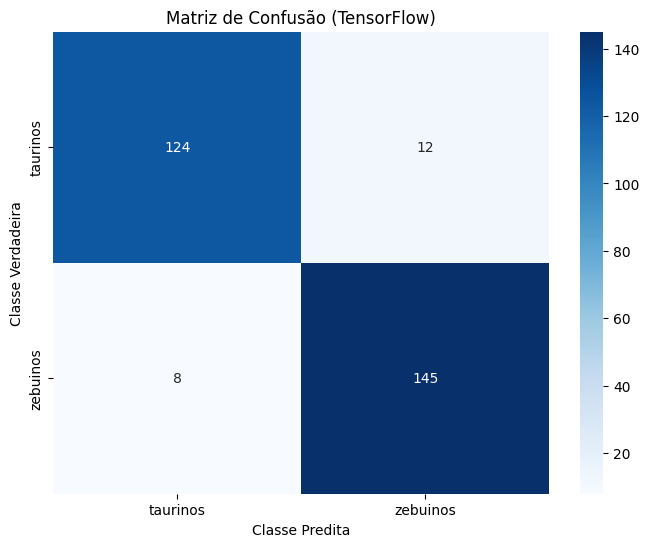


Imagem 2: Matriz de confusão

Na avaliação do modelo com os dados de teste, os resultados permaneceram consistentes, apresentando precisão de 93,08%, recall de 93,08% e loss de 0,218%. O F1-Score foi bem equilibrado entre as classes, com 92,5% para zebuínos e 93,5% para taurinos, evidenciando que o modelo é eficaz em diferenciar entre as duas subespécies.

A análise da matriz de confusão revelou que, entre os 289 exemplos de teste, 124 amostras de zebuínos foram corretamente classificadas, enquanto 12 amostras de zebuínos foram incorretamente classificadas como taurinos. Por outro lado, 145 amostras de taurinos foram corretamente classificadas, com apenas 8 amostras sendo identificadas erroneamente como zebuínos. Esses dados indicam que os erros são relativamente baixos e que o modelo apresenta um bom equilíbrio na classificação entre as classes.

1. **CONCLUSÃO**

Os resultados demonstram a viabilidade de aplicar inteligência artificial na classificação de bovinos, oferecendo uma alternativa prática e precisa em relação aos métodos tradicionais de identificação. Essa abordagem pode contribuir significativamente para a seleção genética, manejo e estratégias de melhoramento animal, reduzindo custos e tempo em processos operacionais.

Entretanto, algumas limitações devem ser reconhecidas. O leve overfitting identificado no treinamento sugere a possibilidade de melhorias no modelo, como a aplicação de técnicas de regularização, o aumento do conjunto de dados ou ajustes nos hiperparâmetros. Além disso, a ausência de comparações com outros métodos de classificação, como SVM ou Random Forest, limita a contextualização dos resultados obtidos.

Como perspectivas futuras, sugere-se ampliar o escopo do trabalho, testando o modelo em dados de outras populações ou adicionando mais classes ao problema de classificação. Comparar o desempenho do modelo com outras técnicas também pode validar e enriquecer as conclusões obtidas neste estudo. De maneira geral, os resultados apresentados demonstram que o modelo tem potencial para contribuir com soluções automatizadas na produção animal, reforçando a importância da adoção de tecnologias baseadas em inteligência artificial no setor.

**REFERÊNCIAS BIBLIOGRÁFICAS**

ABIEC. Associação Brasileira das Indústrias Exportadoras de Carne. **Relatório Anual de Produção e Exportação de Carnes do Brasil - Beef Report. 2022.** Disponível em : https://www.abiec.com.br/publicacoes/beef-report-2022/ . Acesso: 142/12/2024.

LIMA JÚNIOR, D. M. Alguns aspectos qualitativos da carne bovina: uma revisão. **Acta Veterinaria Brasilica**, v.5, n.4, p.351-358, 2011.

MENDONÇA, K. A. et al. Importance of biometric aspects in breeding selection. [**PUBVET**](https://www.cabidigitallibrary.org/action/doSearch?do=PUBVET), 2019, Vol. 13, n°. 12.

NETO, J. A. S. Distúrbios metabólicos em ruminantes - Uma Revisão. **Dialnet**. vol. 8, n° 4, 2014. pág. 157-186.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. In Proceedings of the International Conference on Machine Learning (ICML), 2014.