UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
INE-DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
INE5430 - INTELIGÊNCIA ARTIFICIAL

PEDRO HENRIQUE TAGLIALENHA (22203674)
MATHEUS FERNANDES BIGOLIN (22200371)

GATO VS NÃO-GATO

FLORIANÓPOLIS
2025

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento de padrões, um campo fundamental da inteligência artificial, capacita máquinas a identificar e interpretar informações complexas a partir de dados brutos. Uma de suas aplicações mais proeminentes é a visão computacional, onde redes neurais são treinadas para "enxergar" e entender o conteúdo de imagens.

O presente trabalho aborda um problema de visão computacional: a classificação binária de imagens para distinguir entre "gatos" e "não gatos". Foram implementados e avaliados quatro modelos distintos: um Perceptron Simples (Regressão Logística), uma Rede Neural Rasa, uma Rede Neural Convolucional (CNN) básica e, por fim, uma CNN Otimizada com técnicas de regularização para combater o overfitting.

Este relatório detalha o processo de pré-processamento dos dados, a arquitetura de cada modelo, a análise comparativa dos resultados obtidos, incluindo métricas de acurácia, matriz de confusão e curvas de aprendizado, e, por fim, as conclusões sobre a eficácia de cada abordagem para o problema proposto.

2 PREPARAÇÃO DOS DADOS

O pré-processamento seguiu as etapas padrão para problemas de visão computacional:

- Carregamento: Os dados foram extraídos dos arquivos train_catvnoncat.h5 e test_catvnoncat.h5.
- Informações do Dataset:
    - Conjunto de Treino: 209 imagens.
    - Conjunto de Teste: 50 imagens (distribuídas em 17 "Não Gato" e 33 "Gato").
    - Dimensão das Imagens: 64x64 pixels com 3 canais de cor (RGB).
- Normalização: Os valores dos pixels foram escalados do intervalo [0, 255] para [0, 1] dividindo-os por 255.
- Redimensionamento (Flattening): Para os modelos de Regressão Logística e Rede Rasa, as imagens foram achatadas em um vetor de 12.288 features. A estrutura 2D original foi mantida para as CNNs.

3 ESCOLHAS DE PARÂMETROS

A construção de um modelo de deep learning envolve a definição de inúmeros hiperparâmetros. As escolhas feitas neste trabalho foram baseadas em práticas consolidadas na comunidade de machine learning e adaptadas à natureza específica do problema. A seguir, detalhamos e justificamos as principais escolhas.

3.1 Parâmetros de Treinamento (Comuns a todos os modelos)

Otimizador: Adam (Adaptive Moment Estimation)

- Todos os modelos foram compilados com o otimizador Adam.
- Adam é um otimizador amplamente utilizado que combina as vantagens de dois outros métodos populares: AdaGrad e RMSProp. Ele adapta a taxa de aprendizado para cada parâmetro individualmente, o que geralmente leva a uma convergência mais rápida e estável do que o Gradiente Descendente Estocástico (SGD) tradicional.

- **Taxa de Aprendizado (Learning Rate)**
  - **Escolha:** Foi utilizada a taxa de aprendizado padrão do otimizador Adam no Keras/TensorFlow, que é 0.001.
- **Função de Perda (Loss Function): binary_crossentropy**
  - A função de perda binary_crossentropy foi usada em todos os experimentos.
  - Como o problema é uma classificação binária (Gato vs. Não Gato), esta é a função de perda matematicamente apropriada. Ela mede a "distância" entre a distribuição de probabilidade prevista pelo modelo (a saída da camada sigmoid) e a distribuição verdadeira (o label 0 ou 1), penalizando previsões incorretas de forma logarítmica.
- **Tamanho do Lote (Batch Size): 32**
  - O treinamento foi realizado com um batch_size de 32.
  - Um tamanho de lote de 32 é um padrão de fato na área. Ele representa um excelente equilíbrio entre:
    1. **Estabilidade do Gradiente:** É grande o suficiente para fornecer uma estimativa estável do gradiente da função de perda.
    2. **Eficiência Computacional:** Lotes de tamanho moderado aproveitam bem a paralelização do hardware moderno (GPUs).
    3. **Efeito de Regularização:** Lotes menores que o dataset completo introduzem um certo ruído no processo de treinamento, o que pode ajudar a evitar mínimos locais ruins e a generalizar melhor.

3.2 Parâmetros de Arquitetura

- **Função de Ativação (Camadas Ocultas): ReLU (Rectified Linear Unit)**
  - ReLU foi utilizada em todas as camadas ocultas (densas e convolucionais).
  - ReLU (f(x) = max(0, x)) por duas razões principais:
    1. **Eficiência Computacional:** É extremamente rápida de calcular.
    2. **Mitigação do Desvanecimento do Gradiente:** Ao contrário da sigmoid ou tanh, que saturam (têm gradiente próximo de zero) para

valores de entrada grandes, a ReLU tem um gradiente constante para todas as entradas positivas. Isso facilita o treinamento de redes mais profundas.

- **Função de Ativação (Camada de Saída): Sigmoid**
  - A camada final de todos os modelos utilizou a função sigmoid.
  - Para uma classificação binária, a função sigmoid é essencial. Ela mapeia qualquer valor de entrada para o intervalo [0, 1], o que permite que a saída da rede seja interpretada como a probabilidade de a amostra pertencer à classe positiva (neste caso, "Gato").
- **Parâmetros da CNN (Modelo 3 e 4)**
  - **Tamanho do Kernel:** (3, 3). As camadas convolucionais utilizaram filtros de 3x3.
  - **Pooling:** MaxPooling2D com pool_size=(2, 2).

3.3 Parâmetros de Regularização (Modelo 4: CNN Otimizada)

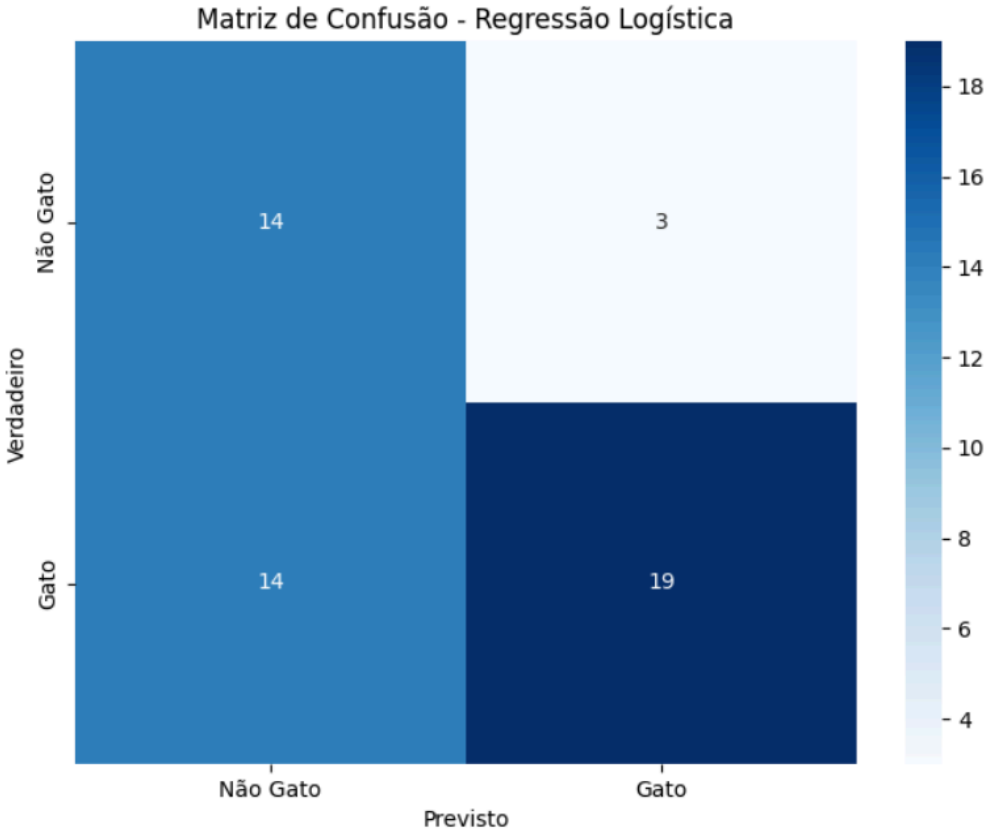
- **Taxa de Dropout: 0.5**
  - Uma camada Dropout com uma taxa de 0.5 foi adicionada.
  - Dropout é uma técnica de regularização poderosa que "desliga" aleatoriamente uma fração dos neurônios durante cada etapa do treinamento. Isso impede que a rede se torne excessivamente dependente de neurônios específicos, forçando-a a aprender representações mais distribuídas e robustas. Uma taxa de 0.5 é um valor agressivo, escolhido para combater o forte overfitting observado no Modelo 3.
- **Early Stopping: patience=10**
  - O treinamento foi monitorado com base na val_loss e configurado para parar se não houvesse melhora por 10 épocas consecutivas.
  - Em vez de treinar por um número fixo de épocas e arriscar parar muito cedo (underfitting) ou muito tarde (overfitting), o EarlyStopping permite que o treinamento continue enquanto o modelo estiver melhorando em dados não vistos. Uma paciência (patience) de 10 épocas foi escolhida para dar ao modelo uma chance de se recuperar de flutuações aleatórias na performance de validação antes de concluir que a convergência foi atingida. A opção restore_best_weights=True garante que o modelo final retido seja aquele do pico de seu desempenho, e não o da última época de treinamento.

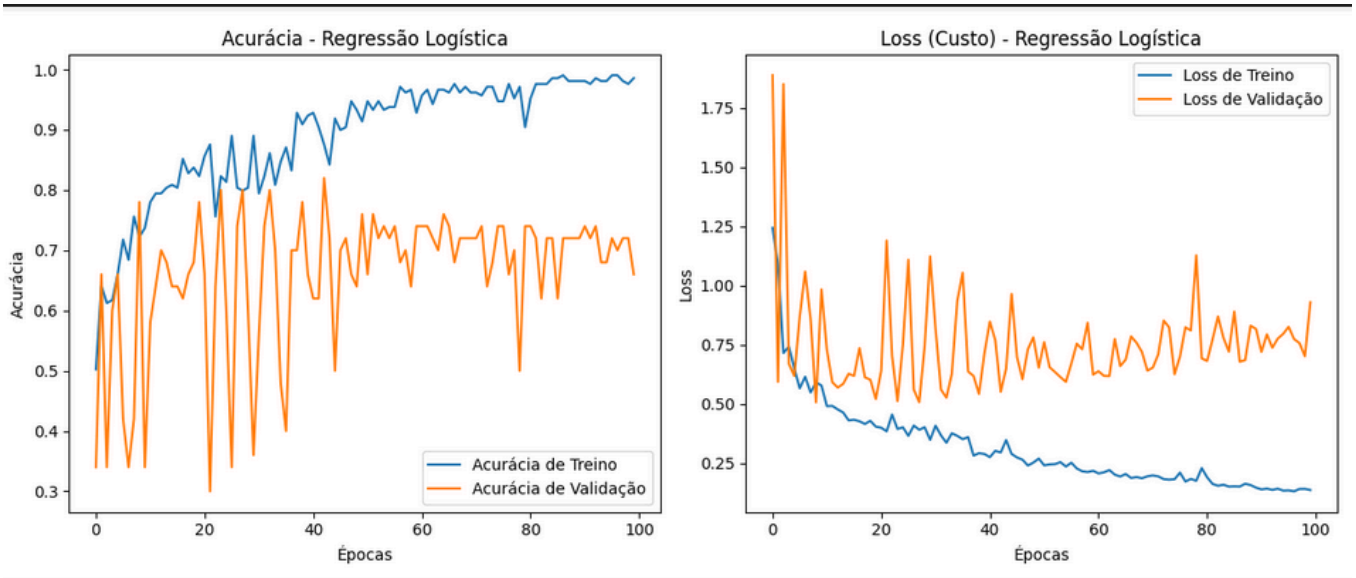
4 ARQUITETURA E ANÁLISE DOS EXPERIMENTOS

Foram realizados quatro experimentos para avaliar a performance de diferentes arquiteturas.

4.1 Modelo 1: Regressão Logística (Perceptron Simples)

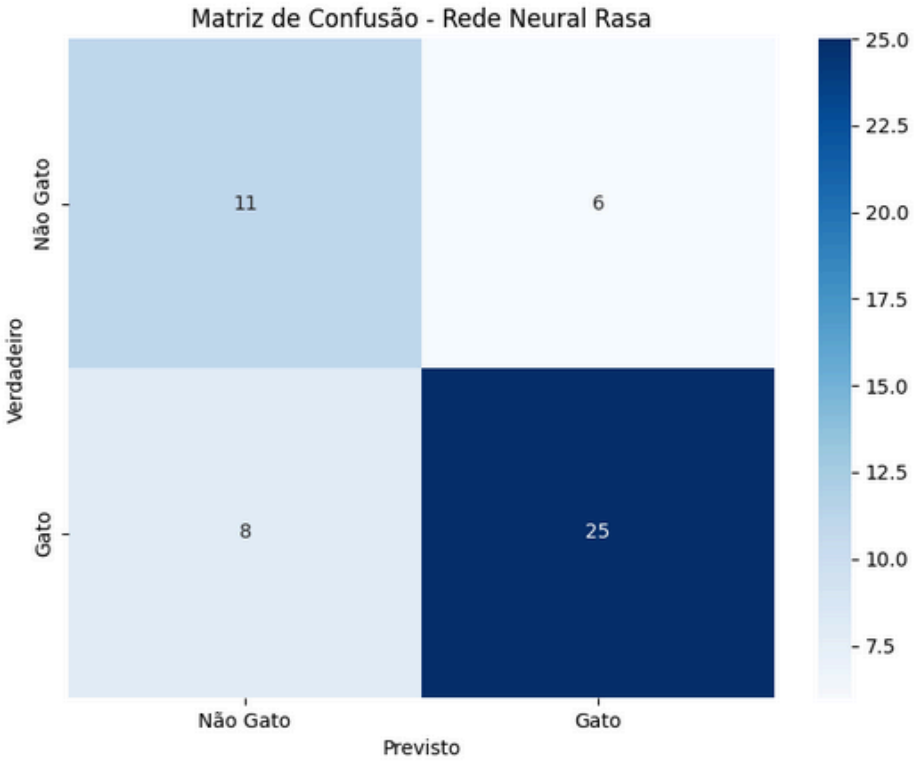
- Arquitetura: Uma única camada densa com 1 neurônio e ativação sigmoid.
- Acurácia no Teste: 66.00%
    - Matriz de Confusão: O modelo classificou incorretamente 14 imagens de "Gato" como "Não Gato" (Falsos Negativos), indicando uma forte dificuldade em identificar a classe positiva. Ele acertou 19 de 33 gatos e 14 de 17 não gatos.
    - Curvas de Aprendizado: As curvas mostram um overfitting severo. A acurácia de treino se aproxima de 100%, enquanto a de validação permanece volátil e em um patamar baixo (~60-70%). A loss de validação é extremamente ruidosa. Isso demonstra que um modelo linear simples não consegue capturar a complexidade do problema e, ao invés de aprender, apenas memoriza os dados de treino.

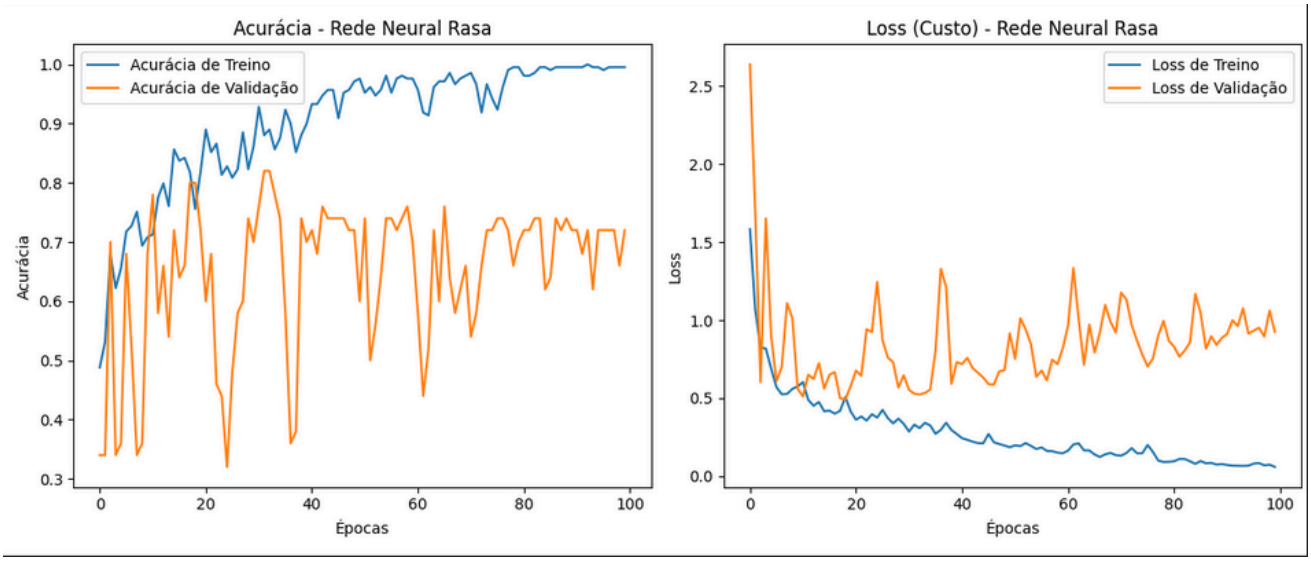




4.2. Modelo 2: Rede Neural Rasa (1 Camada Oculta)

- Arquitetura: Uma camada oculta com 32 neurônios (ReLU) e uma camada de saída com 1 neurônio (sigmoid).
- Acurácia no Teste: 72.00%
    - Matriz de Confusão: Houve uma melhora em relação à Regressão Logística. O modelo acertou 25 de 33 gatos e 11 de 17 não gatos. O aumento da capacidade do modelo (393.281 parâmetros treináveis) permitiu aprender features mais complexas.
    - Curvas de Aprendizado: Apesar da melhora na acurácia, o problema do overfitting persiste de forma crítica. O gap entre a acurácia de treino (quase 100%) e a de validação é enorme. O modelo tem capacidade suficiente para decorar perfeitamente o conjunto de treino, mas falha em generalizar esse conhecimento para os dados de teste



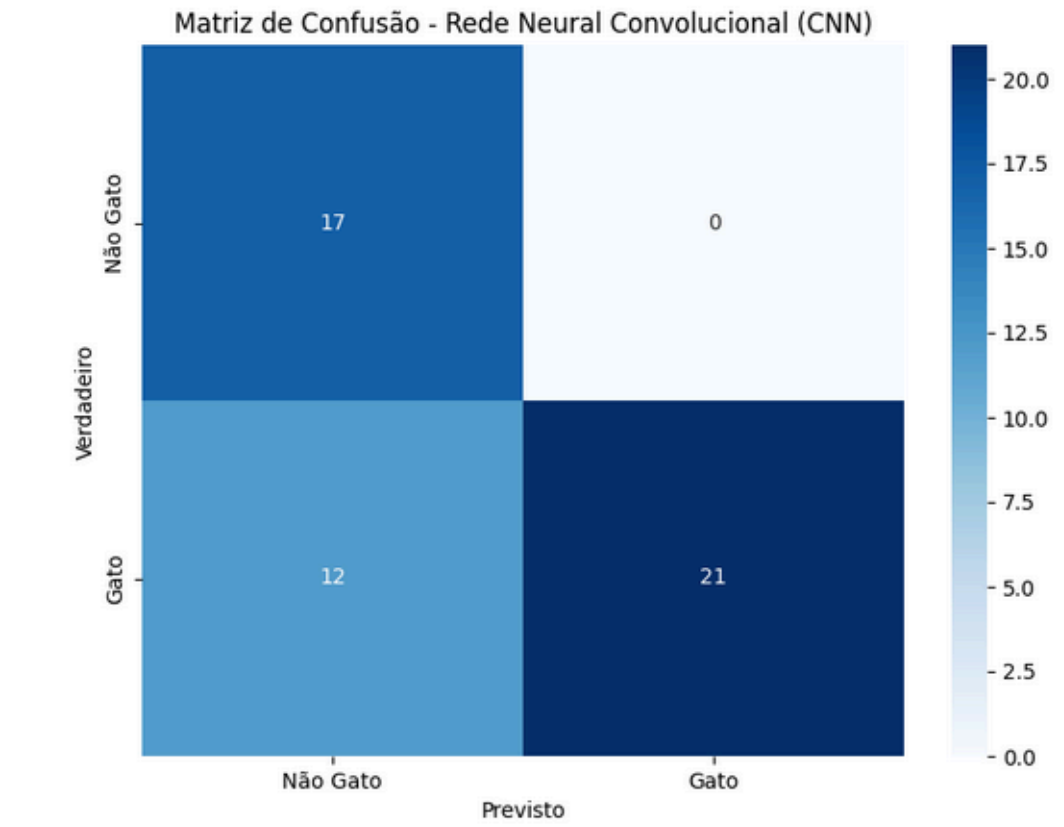


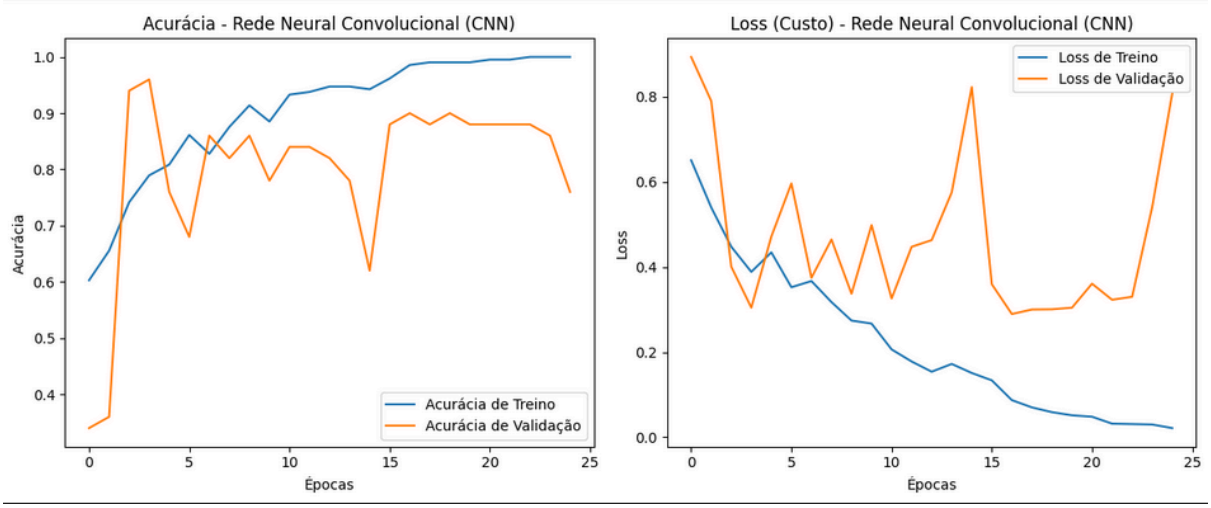
4.3. Modelo 3: Rede Neural Convolucional (CNN)

- Arquitetura: Duas camadas convolucionais com MaxPooling, seguidas por uma camada densa de 64 neurônios e a camada de saída. O modelo possui 822.337 parâmetros.
- Acurácia no Teste: 76.00%
    - Matriz de Confusão: Este resultado é particularmente interessante. O modelo identificou corretamente todas as 17 imagens de "Não Gato" (recall de 100% para essa classe). No entanto, foi extremamente conservador, classificando

12 imagens de "Gato" como "Não Gato". Isso resultou em um recall baixo (64%) para a classe "Gato". O modelo aprendeu a não cometer erros ao prever "Não Gato", mas ao custo de errar muitos gatos.

- Curvas de Aprendizado: As curvas revelam um comportamento clássico de overfitting em CNNs sem regularização. A acurácia de validação sobe bem nas primeiras épocas, mas depois se torna errática e cai. A loss de validação atinge um mínimo por volta da época 12 e depois dispara, indicando que o modelo começou a "desaprender" features generalizáveis e a memorizar o ruído dos dados de treino.



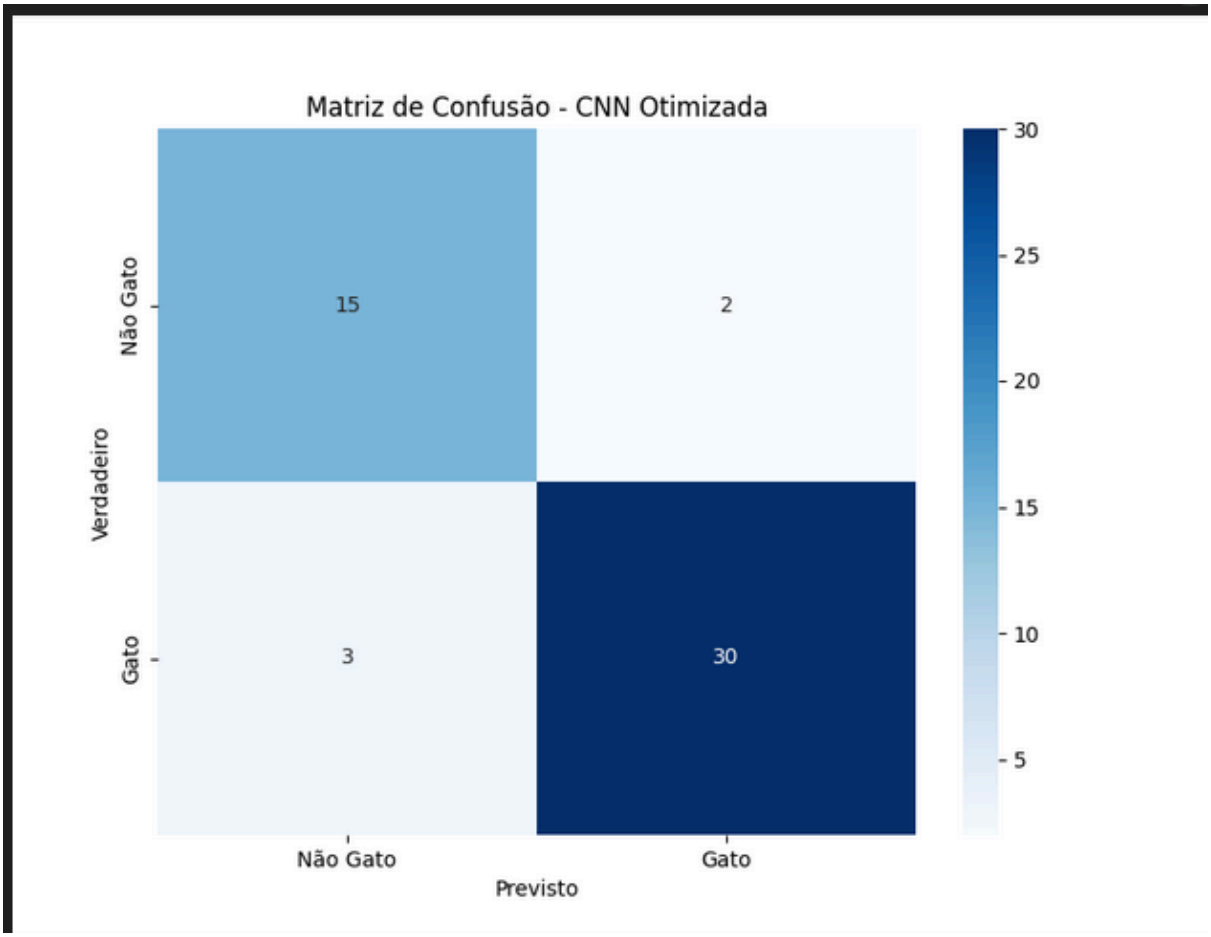


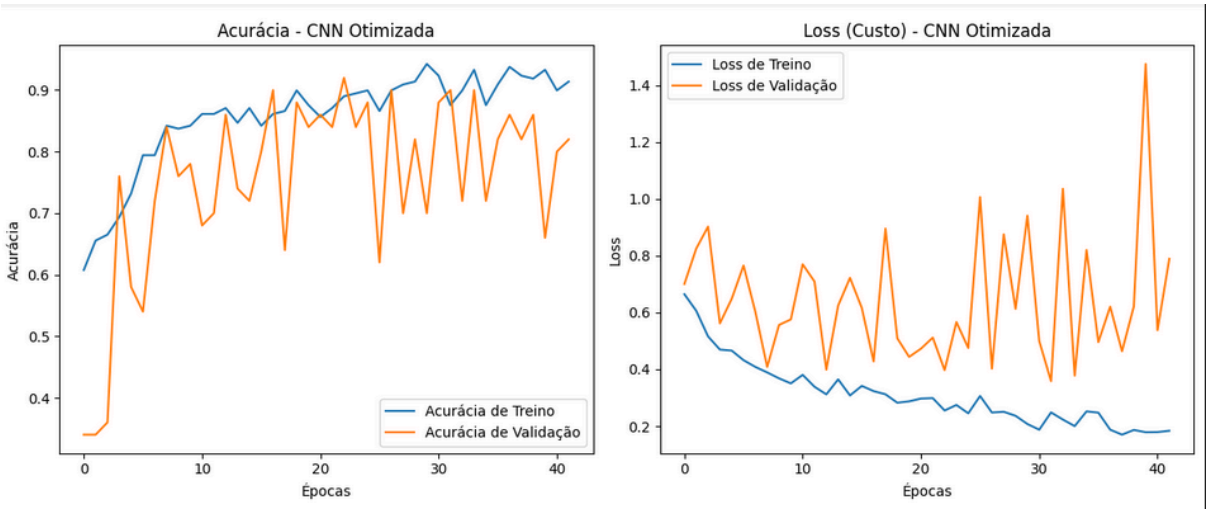
4.4. Modelo 4: CNN Otimizada (com Regularização)

Para combater o overfitting observado no modelo anterior, esta arquitetura foi aprimorada com duas técnicas principais:

1. Aumento de Dados (Data Augmentation): Camadas de RandomFlip e RandomRotation foram adicionadas para criar variações das imagens de treino em tempo real.
2. Dropout: Uma camada de Dropout(0.5) foi inserida antes da camada de saída, "desligando" 50% dos neurônios aleatoriamente durante o treino para forçar a rede a aprender features mais robustas.
3. Early Stopping: O treinamento foi configurado para parar automaticamente se a loss de validação não melhorasse por 10 épocas, restaurando os pesos do melhor modelo.

- Arquitetura: CNN com Data Augmentation, Dropout e 802.949 parâmetros.
- Acurácia no Teste: 90.00%
    - Matriz de Confusão: O resultado é um salto expressivo na performance. O modelo agora é muito mais equilibrado, acertando 30 de 33 gatos e 15 de 17 não gatos. O número de erros totais caiu de 12 (na CNN base) para apenas 5.
    - Curvas de Aprendizado: As curvas são a prova do sucesso da regularização. A distância entre as curvas de treino e validação (tanto de acurácia quanto de loss) é drasticamente menor. Embora ainda exista ruído (esperado para um dataset pequeno), as curvas seguem uma tendência muito mais próxima e saudável, indicando que o modelo está generalizando muito melhor. O EarlyStopping encerrou o treinamento na época 42, evitando o overfitting que certamente ocorreria se o treino continuasse.


Matriz de Confusão - CNN Otimizada
Não Gato
Gato
Verdadeiro
15
2
3
30
Não Gato
Gato
Previsto
30
25
20
15
10
5


Acurácia - CNN Otimizada
Acurácia
Épocas
Acurácia de Treino
Acurácia de Validação
Loss (Custo) - CNN Otimizada
Loss
Épocas
Loss de Treino
Loss de Validação

5 ANÁLISE E CONCLUSÃO

Modelo	Acurácia no Teste	Principal Observação
1. Regressão Logística	66.00%	Overfitting severo e performance limitada.
2. Rede Neural Rasa	72.00%	Leve melhora, mas ainda dominado por overfitting extremo.
3. CNN Base	76.00%	Aprende features espaciais, mas overfita rapidamente, gerando um modelo enviesado.
4. CNN Otimizada	90.00%	Regularização mais eficaz, combatendo o overfitting e alcançando o melhor e mais equilibrado resultado.

Este trabalho demonstrou a importância da arquitetura e da regularização no treinamento de redes neurais. Os modelos lineares e de rede rasa, operando em dados achatados, mostraram-se inadequados e sucumbiram ao overfitting. A introdução da Rede Neural Convolucional (CNN) representou um salto qualitativo, pois sua arquitetura é inerentemente capaz de aprender as características espaciais das imagens.

No entanto, o experimento mais valioso foi a comparação entre a CNN base e a CNN otimizada. Ficou evidente que apenas aumentar a complexidade do modelo não garante um bom resultado, especialmente com um conjunto de dados limitado. As técnicas de Data Augmentation e Dropout foram cruciais para controlar o overfitting.