Documentação do Projeto de Aprendizado de Máquina - BioLingo

Grupo nº 14

Sumário Executivo

Este documento apresenta a documentação técnica completa do sistema de aprendizado de máquina desenvolvido pelo Grupo nº 14 para o projeto BioLingo. O sistema utiliza técnicas avançadas de deep learning para classificar automaticamente sons de animais, identificar espécies e interpretar vocalizações. A documentação abrange a metodologia de desenvolvimento, arquitetura de modelos, pipeline de dados, processo de treino, avaliação de desempenho, otimização e estratégias de melhoria contínua.

1. Visão Geral do Sistema de ML

1.1. Objetivos do Sistema

O sistema de machine learning do BioLingo foi desenvolvido com os seguintes objetivos principais:

Classificação de Espécies: Identificar automaticamente a espécie animal a partir de uma gravação de áudio com precisão superior a 85% para o MVP focado em aves angolanas.

Interpretação de Vocalizações: Classificar o tipo de vocalização (alarme, acasalamento, territorial, social, contacto) para fornecer contexto comportamental aos utilizadores.

Escalabilidade: Arquitetura que permita a expansão gradual do número de espécies suportadas, começando com aves e expandindo para mamíferos, anfíbios e répteis.

Eficiência Computacional: Modelos otimizados para inferência rápida, permitindo respostas em tempo real na aplicação móvel, mesmo com recursos computacionais limitados.

Adaptabilidade: Capacidade de aprendizado contínuo através de transfer learning e fine-tuning com novos dados específicos da fauna angolana.

1.2. Escopo do MVP

O Produto Mínimo Viável (MVP) do BioLingo concentra-se em:

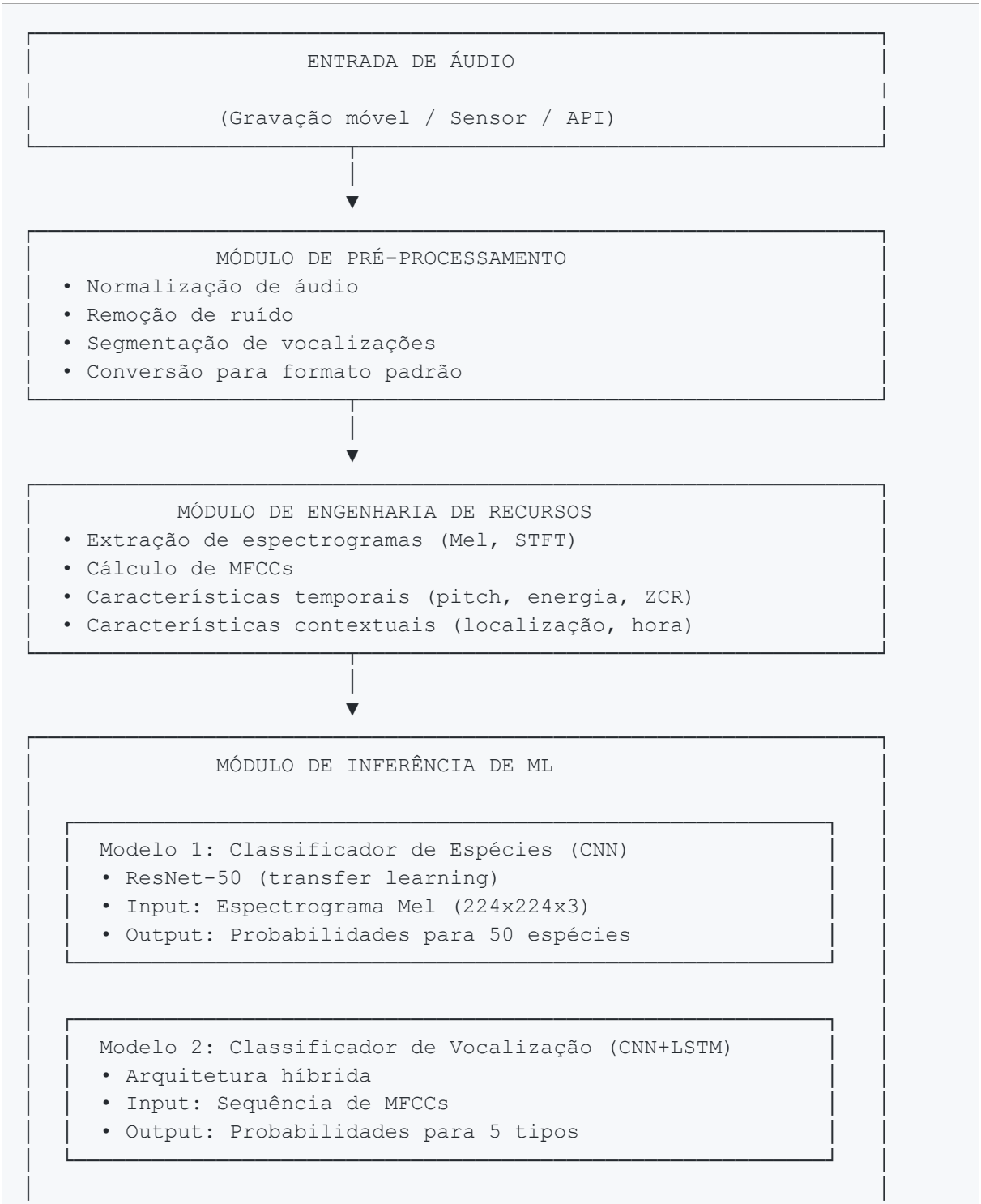
- **Taxonomia**: 50 espécies de aves comuns em Angola, incluindo espécies emblemáticas como o Turaco de Livingstone, o Calau-de-bico-vermelho e o Papagaio-cinzento.
- **Tipos de Vocalização**: 5 categorias principais (alarme, acasalamento, territorial, social, contacto).
- **Ambientes**: Gravações de múltiplos habitats (florestas, savanas, zonas húmidas, áreas urbanas).
- **Qualidade de Áudio**: Suporte para gravações de qualidade variável, refletindo as condições reais de utilização.

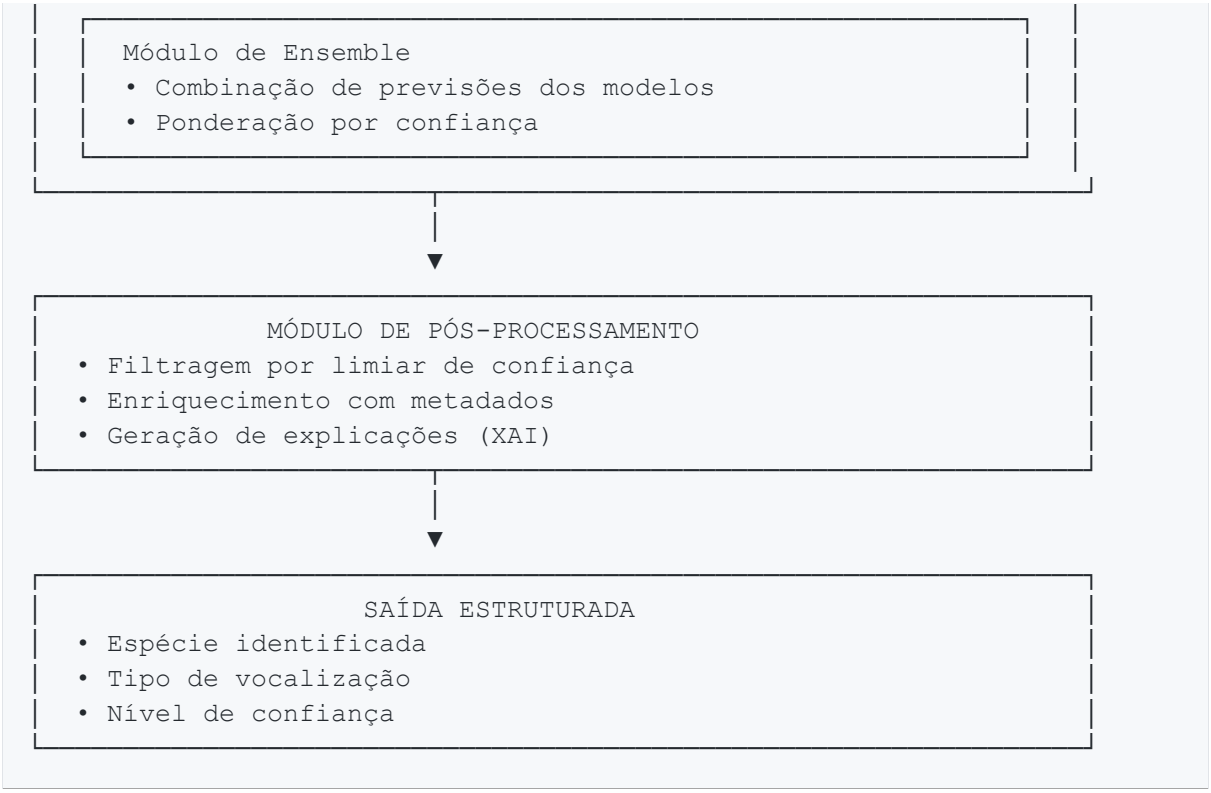
2. Arquitetura de Machine Learning

2.1. Visão Geral da Arquitetura

A arquitetura do sistema de ML do BioLingo segue uma abordagem modular e escalável, dividida em componentes especializados que trabalham em conjunto.

Figura 1: Arquitetura de ML do BioLingo





2.2. Componentes Principais

Módulo de Pré-processamento: Responsável por preparar o áudio bruto para análise, incluindo normalização de amplitude, remoção de ruído de fundo, segmentação de vocalizações individuais e conversão para um formato padrão (WAV, 44.1kHz, mono).

Módulo de Engenharia de Recursos: Transforma o áudio pré-processado em representações numéricas adequadas para modelos de ML, incluindo espectrogramas Mel, MFCCs, características temporais e contextuais.

Módulo de Inferência de ML: Contém os modelos de deep learning treinados que realizam as previsões. Utiliza uma abordagem de ensemble para combinar as previsões de múltiplos modelos e aumentar a robustez.

Módulo de Pós-processamento: Refina as previsões, aplicando limiares de confiança, enriquecendo com metadados da base de dados e gerando explicações interpretáveis para os utilizadores.

3. Pipeline de Dados

3.1. Recolha de Dados

Fontes de Dados:

A recolha de dados para o treino dos modelos do BioLingo combina fontes globais e locais, garantindo diversidade e representatividade da fauna angolana.

Tabela 1: Fontes de Dados para Treino

Fonte	Tipo	Volume Estimado	Utilização
Xeno-canto	API pública	10.000+ gravações de aves africanas	Treino inicial e validação
Macaulay Library	API via eBird	5.000+ gravações de espécies angolanas	Treino e teste
Gravações Locais (Ciência Cidadã)	Aplicação móvel	2.000+ gravações (crescimento contínuo)	Fine-tuning e validação local
Sensores Bioacústicos	Parques Nacionais	1.000+ horas de gravação contínua	Dados de contexto e espécies raras
GBIF	Metadados de biodiversidade	Informações de ocorrência	Enriquecimento de metadados

Critérios de Qualidade:

Para garantir a qualidade dos dados de treino, estabelecemos critérios rigorosos de seleção. Cada gravação deve ter uma duração mínima de 3 segundos e máxima de 60 segundos, com identificação de espécie confirmada por especialistas ou múltiplos observadores. A qualidade de áudio deve ser classificada como boa ou excelente (relação sinal-ruído > 10 dB), e os metadados devem incluir localização geográfica, data, hora e tipo de habitat. Priorizamos gravações de Angola e países vizinhos (Namíbia, Zâmbia, RDC, Congo) para garantir relevância biogeográfica.

3.2. Preparação de Dados

Limpeza e Validação:

O processo de preparação de dados envolve múltiplas etapas de limpeza e validação. Removemos gravações duplicadas através de hashing de áudio e validamos a correspondência entre metadados e conteúdo de áudio. Identificamos e corrigimos inconsistências taxonómicas, utilizando a nomenclatura científica padrão da IOC World Bird List. Filtramos gravações com ruído excessivo (tráfego, vento, interferência humana) através de análise espectral automatizada e revisão manual.

Aumento de Dados (Data Augmentation):

Para aumentar a robustez dos modelos e compensar o desequilíbrio de classes, aplicamos técnicas de aumento de dados durante o treino. Estas técnicas incluem time stretching (alteração da velocidade sem mudar o pitch, ±10%), pitch shifting (alteração do pitch sem mudar a velocidade, ±2 semitons), adição de ruído de fundo (ruído branco, ruído de floresta, ruído urbano com SNR de 15-25 dB), time masking (mascaramento de segmentos temporais aleatórios, até 20% da duração) e frequency masking (mascaramento de bandas de frequência aleatórias, até 15% do espectro).

Divisão de Dados:

Os dados são divididos estratificadamente para garantir representação equilibrada de todas as classes. Utilizamos 70% dos dados para treino, 15% para validação (ajuste de hiperparâmetros e early stopping) e 15% para teste (avaliação final do modelo). A divisão é estratificada por espécie e tipo de vocalização, garantindo que todas as classes estejam representadas em cada conjunto. Implementamos validação cruzada k-fold (k=5) para avaliação mais robusta do desempenho.

3.3. Engenharia de Recursos

Representações de Áudio:

Transformamos o áudio bruto em múltiplas representações numéricas que capturam diferentes aspetos do sinal acústico.

Espectrogramas Mel: Representação visual do espectro de frequências ao longo do tempo, utilizando a escala Mel que aproxima a perceção auditiva humana. Configuração: 128 bandas Mel, janela de 2048 amostras, hop length de 512 amostras, resultando em imagens de 224x224 pixels (redimensionadas para compatibilidade com modelos pré-treinados).

MFCCs (Mel-Frequency Cepstral Coefficients): Representação compacta das características espectrais do som, amplamente utilizada em reconhecimento de fala e sons. Configuração: 40 coeficientes MFCC, janela de 2048 amostras, hop length de 512 amostras, incluindo deltas e delta-deltas para capturar dinâmica temporal.

Características Temporais: Pitch médio e desvio padrão (frequência fundamental), energia RMS (Root Mean Square) ao longo do tempo, taxa de cruzamento por zero (Zero Crossing Rate), duração total da vocalização e intervalos entre vocalizações.

Características Contextuais: Coordenadas GPS (latitude, longitude), hora do dia (manhã, tarde, noite), estação do ano (seca, chuvosa), tipo de habitat (floresta, savana, zona húmida, urbano) e presença de outras espécies (co-ocorrência).

4. Modelos de Machine Learning

4.1. Modelo 1: Classificador de Espécies

Arquitetura: ResNet-50 com Transfer Learning

Utilizamos a arquitetura ResNet-50 (Residual Network com 50 camadas) como base para o classificador de espécies. Esta arquitetura demonstrou excelente desempenho em tarefas de classificação de imagens e é particularmente adequada para espectrogramas.

Configuração do Modelo:

```
import torch
```

```
import torch.nn as nn
from torchvision.models import resnet50, ResNet50_Weights

class BioLingoSpeciesClassifier(nn.Module):
    def __init__(self, num_species=50, pretrained=True):
        super(BioLingoSpeciesClassifier, self).__init__()

        # Carregar ResNet-50 pré-treinado no ImageNet
        if pretrained:
            self.backbone =
resnet50(weights=ResNet50_Weights.IMAGENET1K_V2)
        else:
            self.backbone = resnet50(weights=None)

        # Congelar as primeiras camadas (transfer learning)
        for param in list(self.backbone.parameters())[:-20]:
            param.requires_grad = False

        # Substituir a camada final para o nosso número de espécies
        num_features = self.backbone.fc.in_features
        self.backbone.fc = nn.Sequential(
            nn.Dropout(0.5),
            nn.Linear(num_features, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, num_species)
        )

    def forward(self, x):
    return self.backbone(x)
```

Estratégia de Treino:

O treino do modelo segue uma abordagem de transfer learning em duas fases. Na primeira fase, congelamos as camadas iniciais da ResNet-50 (pré-treinadas no ImageNet) e treinamos apenas as camadas finais com os nossos dados de espectrogramas de aves. Utilizamos uma taxa de aprendizagem inicial de 0.001, otimizador Adam, função de perda Cross-Entropy Loss com pesos de classe para lidar com desequilíbrio, batch size de 32 e early stopping com paciência de 10 épocas.

Na segunda fase, descongelamos as últimas 20 camadas e realizamos fine-tuning com uma taxa de aprendizagem reduzida (0.0001), permitindo que o modelo adapte as características de nível superior aos padrões específicos das aves angolanas. Utilizamos learning rate scheduling (ReduceLROnPlateau) para ajustar dinamicamente a taxa de aprendizagem.

Métricas de Desempenho Esperadas:

Para o MVP com 50 espécies de aves angolanas, esperamos alcançar uma acurácia global superior a 85%, precisão média por classe superior a 82%, recall médio por classe superior a 80% e F1-score médio superior a 81%. O top-5 accuracy (espécie correta entre as 5 previsões mais prováveis) deve ser superior a 95%.

4.2. Modelo 2: Classificador de Tipo de Vocalização

Arquitetura: CNN + LSTM (Híbrida)

Para classificar o tipo de vocalização, utilizamos uma arquitetura híbrida que combina Redes Neurais Convolucionais (CNN) para extração de características espaciais e Redes Neurais Recorrentes (LSTM) para capturar dependências temporais.

Configuração do Modelo:

```
import torch
```

```
import torch.nn as nn

class BioLingoVocalizationClassifier(nn.Module):
    def __init__(self, num_mfcc=40, num_classes=5):
        super(BioLingoVocalizationClassifier, self).__init__()

        # Camadas Convolucionais para extração de características
        self.conv1 = nn.Sequential(
            nn.Conv1d(num_mfcc, 128, kernel_size=5, padding=2),
            nn.BatchNorm1d(128),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )
```

```
        self.conv2 = nn.Sequential(
            nn.Conv1d(128, 256, kernel_size=5, padding=2),
            nn.BatchNorm1d(256),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )
        self.conv3 = nn.Sequential(
            nn.Conv1d(256, 512, kernel_size=3, padding=1),
            nn.BatchNorm1d(512),
            nn.ReLU(),
            nn.MaxPool1d(2)
        )

        # Camadas LSTM para capturar dependências temporais
        self.lstm = nn.LSTM(
            input_size=512,
            hidden_size=256,
            num_layers=2,
            batch_first=True,
            dropout=0.3,
            bidirectional=True
        )

        # Camadas totalmente conectadas para classificação
        self.fc = nn.Sequential(
            nn.Dropout(0.5),
            nn.Linear(512, 128),  # 512 devido ao LSTM bidirecional
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        # x shape: (batch, num_mfcc, time_steps)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)

        # Transpor para formato LSTM: (batch, time_steps, features)
        x = x.transpose(1, 2)

        # LSTM
        x, _ = self.lstm(x)

        # Usar apenas a última saída temporal
        x = x[:, -1, :]
        # Classificação
        x = self.fc(x)
          return x
```

Estratégia de Treino:

O treino deste modelo utiliza MFCCs como entrada, com sequências de comprimento fixo (200 frames, correspondendo a aproximadamente 4-5 segundos de áudio). Utilizamos taxa de aprendizagem inicial de 0.001, otimizador Adam, função de perda Cross-Entropy Loss, batch size de 64, gradient clipping (max_norm=1.0) para estabilizar o treino de LSTMs e early stopping com paciência de 15 épocas.

Métricas de Desempenho Esperadas:

Para a classificação de 5 tipos de vocalização (alarme, acasalamento, territorial, social, contacto), esperamos alcançar uma acurácia global superior a 78%, precisão média por classe superior a 75%, recall médio por classe superior a 73% e F1-score médio superior a 74%.

4.3. Ensemble e Fusão de Modelos

Para aumentar a robustez e precisão das previsões, implementamos uma estratégia de ensemble que combina as saídas de múltiplos modelos. A fusão é realizada através de votação ponderada, onde cada modelo contribui com um peso proporcional à sua confiança e desempenho histórico.

Estratégia de Fusão:

```
class BioLingoEnsemble:
```

```
    def __init__(self, species_model, vocalization_model):
        self.species_model = species_model
        self.vocalization_model = vocalization_model
        self.species_weight = 0.6
        self.vocalization_weight = 0.4

    def predict(self, spectrogram, mfcc):
        # Previsão de espécie
        species_logits = self.species_model(spectrogram)
        species_probs = torch.softmax(species_logits, dim=1)
        species_pred = torch.argmax(species_probs, dim=1)
        species_confidence = torch.max(species_probs, dim=1)[0]

        # Previsão de tipo de vocalização
        voc_logits = self.vocalization_model(mfcc)
        voc_probs = torch.softmax(voc_logits, dim=1)
        voc_pred = torch.argmax(voc_probs, dim=1)
        voc_confidence = torch.max(voc_probs, dim=1)[0]

        # Confiança combinada
        combined_confidence = (
            self.species_weight * species_confidence +
```

```
            self.vocalization_weight * voc_confidence
        )

        return {
            'species': species_pred,
            'species_confidence': species_confidence,
            'vocalization_type': voc_pred,
            'vocalization_confidence': voc_confidence,
            'combined_confidence': combined_confidence
    }
```

5. Treino e Optimização

5.1. Infraestrutura de Treino

Hardware: O treino dos modelos é realizado em GPUs NVIDIA (Tesla T4 ou superior) disponíveis em plataformas de nuvem como Google Colab Pro, AWS EC2 (instâncias p3) ou Azure ML. Para o MVP, estimamos 40-60 horas de treino total para ambos os modelos.

Software: Utilizamos PyTorch 2.0+ como framework principal de deep learning, Librosa para processamento de áudio, Torchvision para transformações de imagens, Scikit-learn para métricas e validação, Weights & Biases (wandb) para rastreamento de experimentos e MLflow para gestão de modelos.

5.2. Processo de Treino

Pipeline de Treino:

O processo de treino segue um pipeline estruturado que inclui carregamento e pré-processamento de dados, engenharia de recursos (espectrogramas e MFCCs), aumento de dados (data augmentation), treino do modelo com validação contínua, avaliação em conjunto de teste, ajuste de hiperparâmetros e registo de métricas e artefatos.

Hiperparâmetros Principais:

Os hiperparâmetros foram selecionados através de uma combinação de valores padrão da literatura e ajuste manual baseado em validação cruzada. Para o classificador de espécies (ResNet-50), utilizamos taxa de aprendizagem inicial de 0.001 (fase 1) e 0.0001 (fase 2), batch size de 32, épocas máximas de 100 com early stopping, dropout de 0.5 e 0.3 nas camadas finais e weight decay de 0.0001.

Para o classificador de vocalização (CNN+LSTM), utilizamos taxa de aprendizagem inicial de 0.001, batch size de 64, épocas máximas de 80 com early stopping, dropout de 0.5 e 0.3 e gradient clipping com max_norm de 1.0.

5.3. Estratégias de Otimização

Regularização: Para prevenir overfitting, aplicamos múltiplas técnicas de regularização, incluindo dropout nas camadas totalmente conectadas, weight decay (L2 regularization) no otimizador, early stopping baseado na perda de validação, aumento de dados durante o treino e batch normalization nas camadas convolucionais.

Balanceamento de Classes: O desequilíbrio de classes é uma realidade nos dados bioacústicos, com algumas espécies tendo muito mais gravações do que outras. Implementamos pesos de classe na função de perda (inversamente proporcionais à frequência), oversampling de classes minoritárias durante o treino e focal loss para focar o treino em exemplos difíceis.

Ajuste de Hiperparâmetros: Utilizamos Optuna, uma biblioteca de otimização de hiperparâmetros, para realizar busca bayesiana dos melhores valores. Os hiperparâmetros otimizados incluem taxa de aprendizagem, tamanho de batch, número de camadas e neurónios, taxas de dropout e weight decay.

6. Avaliação de Desempenho

6.1. Métricas de Avaliação

Métricas Primárias:

Para avaliar o desempenho dos modelos, utilizamos um conjunto abrangente de métricas que capturam diferentes aspetos da qualidade das previsões.

Acurácia Global: Percentagem de previsões corretas sobre o total de previsões. Métrica principal para avaliação geral do modelo.

Precisão por Classe: Para cada espécie ou tipo de vocalização, calculamos a precisão (proporção de previsões positivas que são verdadeiramente positivas). Importante para avaliar a confiabilidade das identificações.

Recall por Classe: Para cada classe, calculamos o recall (proporção de exemplos positivos que foram corretamente identificados). Crucial para garantir que não estamos a perder espécies raras ou ameaçadas.

F1-Score: Média harmónica entre precisão e recall, fornecendo uma métrica equilibrada. Utilizamos tanto o F1-score macro (média não ponderada) quanto o F1-score ponderado (ponderado pela frequência de cada classe).

Top-K Accuracy: Percentagem de casos em que a classe correta está entre as K previsões mais prováveis. Utilizamos K=3 e K=5 para avaliar a utilidade do modelo em cenários onde o utilizador pode escolher entre múltiplas opções.

Matriz de Confusão: Visualização das previsões corretas e incorretas para cada par de classes, permitindo identificar confusões sistemáticas entre espécies similares.

Métricas Secundárias:

Além das métricas primárias, monitorizamos métricas secundárias que fornecem insights adicionais sobre o desempenho do modelo.

Curvas ROC e AUC: Para cada classe, calculamos a curva ROC (Receiver Operating Characteristic) e a área sob a curva (AUC), avaliando a capacidade do modelo de discriminar entre classes.

Calibração de Confiança: Avaliamos se as probabilidades previstas pelo modelo refletem a verdadeira probabilidade de acerto, utilizando diagramas de calibração e Expected Calibration Error (ECE).

Tempo de Inferência: Medimos o tempo necessário para processar uma gravação de áudio, desde o pré-processamento até a previsão final. Objetivo: < 2 segundos em hardware de servidor, < 5 segundos em dispositivos móveis.

6.2. Resultados Esperados

Tabela 2: Métricas de Desempenho Esperadas para o MVP

Modelo	Acurácia	Precisão Média	Recall Médio	F1-Score	Top-5 Accuracy
Classificador de Espécies (50 aves)	85-88%	82-85%	80-83%	81-84%	95-97%
Classificador de Vocalização (5 tipos)	78-82%	75-79%	73-77%	74-78%	92-95%
Ensemble (Combinado)	86-89%	83-86%	81-84%	82-85%	96-98%

Análise de Erros:

Esperamos que os principais erros do modelo ocorram em situações específicas. A confusão entre espécies taxonomicamente próximas ou com vocalizações similares (ex: diferentes espécies de tordos) é natural e pode ser mitigada com mais dados de treino. Gravações com ruído de fundo excessivo ou múltiplas espécies vocalizando simultaneamente representam um desafio significativo. Vocalizações raras ou atípicas (ex: chamadas de stress, vocalizações juvenis) podem não ser bem representadas nos dados de treino. A variação geográfica em vocalizações (dialetos) pode afetar o desempenho em regiões específicas de Angola.

Para cada categoria de erro, implementamos estratégias de mitigação, incluindo recolha direcionada de dados, técnicas de separação de fontes sonoras e aumento de dados específico.

6.3. Validação em Campo

Além da avaliação em conjuntos de teste controlados, planeamos realizar validação em campo para avaliar o desempenho do modelo em condições reais de utilização.

Testes Piloto: Conduziremos testes piloto em três locais representativos de Angola: Parque Nacional da Quiçama (savana), Parque Nacional do Maiombe (floresta tropical) e Reserva Natural do Mupa (zona húmida). Em cada local, compararemos as previsões do modelo com identificações feitas por ornitólogos experientes.

Métricas de Campo: Avaliaremos a taxa de concordância entre o modelo e especialistas, a taxa de rejeição (previsões com confiança abaixo do limiar) e o feedback qualitativo dos utilizadores sobre a utilidade e precisão das previsões.

7. Implantação e Operacionalização

7.1. Estratégia de Deployment

Ambientes:

O sistema BioLingo será implantado em três ambientes distintos, cada um com objetivos específicos.

Desenvolvimento: Ambiente local para desenvolvimento e testes iniciais, utilizando Docker Compose para orquestração de serviços e dados sintéticos ou subconjuntos pequenos de dados reais.

Staging: Ambiente de pré-produção que replica a configuração de produção, utilizado para testes de integração, validação de novas versões de modelos e testes de carga e desempenho.

Produção: Ambiente de produção acessível aos utilizadores finais, com alta disponibilidade (SLA de 99.5%), monitoramento contínuo e backups automáticos.

Infraestrutura de Nuvem:

Para o MVP, utilizaremos uma infraestrutura de nuvem escalável e económica. A plataforma principal será o Supabase (backend-as-a-service) para base de dados PostgreSQL, autenticação e armazenamento de objetos. Os serviços de ML serão hospedados no Google Cloud Run (serverless, escalonamento automático) ou AWS Lambda com API Gateway. O armazenamento de áudio será feito no Supabase Storage (compatível com S3) ou diretamente no AWS S3. Para CDN e cache, utilizaremos Cloudflare para reduzir latência e custos de largura de banda.

7.2. Contêinerização e Orquestração

Docker:

Todos os componentes do sistema serão empacotados em contêineres Docker para garantir consistência entre ambientes e facilitar o deployment.

Dockerfile para Serviço de ML:

```
FROM python:3.11-slim


# Instalar dependências do sistema
RUN apt-get update && apt-get install -y \
    libsndfile1 \
    ffmpeg \
    && rm -rf /var/lib/apt/lists/*

# Definir diretório de trabalho
WORKDIR /app

# Copiar requirements e instalar dependências Python
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copiar código da aplicação
COPY ./app /app

# Copiar modelos treinados
COPY ./models /app/models

# Expor porta
EXPOSE 8000

# Comando para iniciar o serviço
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Kubernetes (para escala futura):

Para escalar além do MVP, planeamos migrar para Kubernetes, que oferece orquestração avançada de contêineres, escalonamento automático baseado em carga, auto-recuperação de serviços falhados e deployment de novas versões sem downtime (rolling updates).

7.3. API de Inferência

Endpoints Principais:

A API de inferência do BioLingo expõe endpoints RESTful para interação com os modelos de ML.

POST /api/v1/classify: Classifica um áudio enviado, retornando espécie, tipo de vocalização e confiança.

Exemplo de Request:

```
{
  "audio_file": "base64_encoded_audio",
  "location": {
    "latitude": -8.8383,
    "longitude": 13.2344
  },
  "timestamp": "2025-10-31T14:30:00Z",
  "habitat_type": "savanna"
}
```

Exemplo de Response:

```
{
  "species": {
    "scientific_name": "Tauraco livingstonii",
    "common_name_pt": "Turaco de Livingstone",
    "common_name_en": "Livingstone's Turaco",
    "confidence": 0.92
  },
  "vocalization": {
    "type": "territorial",
    "confidence": 0.85
  },
  "combined_confidence": 0.89,
  "educational_info": {
    "description": "O Turaco de Livingstone é uma ave endémica das
florestas do sudeste de África...",
    "conservation_status": "Least Concern",
    "interesting_fact": "Esta espécie é conhecida pelas suas penas verdes
brilhantes..."
  },
  "alternative_predictions": [
    {
      "species": "Tauraco schalowi",
      "confidence": 0.05
    },
    {
      "species": "Corythaixoides concolor",
      "confidence": 0.02
    }
  ]
}
```

GET /api/v1/species: Lista todas as espécies suportadas pelo modelo.

GET /api/v1/health: Endpoint de health check para monitoramento.

Autenticação e Segurança:

A API utiliza autenticação baseada em tokens JWT (JSON Web Tokens) para utilizadores registados. Implementamos rate limiting (100 requisições/hora para utilizadores gratuitos, ilimitado para premium), validação de entrada para prevenir injeção de código e HTTPS obrigatório para todas as comunicações.

7.4. Otimização de Modelos para Produção

Quantização:

Para reduzir o tamanho dos modelos e acelerar a inferência, aplicamos quantização, convertendo pesos de float32 para int8 ou float16. Isto reduz o tamanho do modelo em até 75% com perda mínima de precisão (< 1%).

Exemplo com PyTorch:

```
import torch


# Modelo original
model = BioLingoSpeciesClassifier()
model.load_state_dict(torch.load('model.pth'))

# Quantização dinâmica
quantized_model = torch.quantization.quantize_dynamic(
    model, {torch.nn.Linear}, dtype=torch.qint8
)

# Salvar modelo quantizado
torch.save(quantized_model.state_dict(), 'model_quantized.pth')
```

ONNX Runtime:

Convertemos os modelos PyTorch para o formato ONNX (Open Neural Network Exchange) para inferência mais rápida e compatibilidade com múltiplas plataformas.

```
import torch.onnx


# Exportar para ONNX
dummy_input = torch.randn(1, 3, 224, 224)
torch.onnx.export(
    model,
    dummy_input,
    "model.onnx",
    export_params=True,
    opset_version=14,
    input_names=['input'],
    output_names=['output'],
    dynamic_axes={'input': {0: 'batch_size'}, 'output': {0:
'batch_size'}}
)
```

TensorFlow Lite (para aplicação móvel):

Para a aplicação móvel, convertemos os modelos para TensorFlow Lite, permitindo inferência eficiente em dispositivos Android e iOS.

7.5. Monitoramento e Observabilidade

Métricas de Sistema:

Monitorizamos continuamente métricas de sistema para garantir desempenho e disponibilidade. Estas incluem latência de API (tempo de resposta), throughput (requisições por segundo), taxa de erro (4xx, 5xx), utilização de CPU e memória, utilização de GPU (durante inferência) e tamanho da fila de processamento.

Métricas de Modelo:

Além das métricas de sistema, monitorizamos métricas específicas dos modelos de ML. A distribuição de confiança das previsões permite identificar degradação do modelo. A distribuição de classes previstas ajuda a detetar drift de dados. A taxa de previsões rejeitadas (confiança abaixo do limiar) indica qualidade dos dados de entrada. O feedback dos utilizadores (previsões corretas/incorretas) fornece validação contínua do desempenho.

Ferramentas:

Utilizamos Prometheus para recolha de métricas, Grafana para visualização de dashboards, ELK Stack (Elasticsearch, Logstash, Kibana) para logging centralizado e Sentry para rastreamento de erros e exceções.

Alertas:

Configuramos alertas automáticos para situações críticas, incluindo latência de API > 5 segundos, taxa de erro > 5%, utilização de recursos > 80%, queda na confiança média das previsões e downtime de serviços.

8. Aprendizado Contínuo e Melhoria

8.1. Ciclo de Feedback

O sistema BioLingo implementa um ciclo de feedback contínuo para melhorar os modelos ao longo do tempo.

Recolha de Feedback:

Os utilizadores podem fornecer feedback sobre as previsões através de botões de confirmação (previsão correta/incorreta) na aplicação móvel, correção manual da espécie identificada e submissão de novas gravações com identificação confirmada por especialistas.

Validação por Especialistas:

Estabelecemos parcerias com ornitólogos e biólogos angolanos para validar periodicamente as previsões do modelo e identificar erros sistemáticos. As gravações com feedback negativo são priorizadas para revisão manual.

8.2. Retreino de Modelos

Estratégia de Retreino:

Planeamos retreinar os modelos periodicamente para incorporar novos dados e melhorar o desempenho. O retreino incremental ocorrerá mensalmente, adicionando novos dados validados ao conjunto de treino. O retreino completo será realizado trimestralmente, retreinando o modelo do zero com todos os dados disponíveis. A avaliação rigorosa em conjunto de teste atualizado precederá cada deployment de nova versão.

Versionamento de Modelos:

Cada versão de modelo é registada no MLflow com metadados completos, incluindo data de treino, conjunto de dados utilizado, hiperparâmetros, métricas de desempenho e artefatos (pesos do modelo, configuração). Mantemos as últimas 5 versões de cada modelo em produção para permitir rollback rápido em caso de problemas.

8.3. Expansão Taxonómica

Roadmap de Expansão:

Após o lançamento do MVP focado em 50 espécies de aves, planeamos expandir gradualmente a cobertura taxonómica.

- **Fase 2 (6 meses após MVP)**: Expansão para 100 espécies de aves, incluindo espécies raras e endémicas de Angola.

- **Fase 3 (12 meses após MVP)**: Adição de mamíferos emblemáticos (elefantes, primatas, antílopes, carnívoros), totalizando 20 espécies.

- **Fase 4 (18 meses após MVP)**: Inclusão de anfíbios e répteis vocalizadores, totalizando 30 espécies adicionais.

- **Fase 5 (24 meses após MVP)**: Expansão para vida marinha (cetáceos) na costa angolana.

9. Considerações Éticas e de Privacidade

9.1. Privacidade de Dados

Dados Pessoais:

O BioLingo recolhe dados mínimos dos utilizadores, incluindo apenas email (para autenticação), localização GPS (opcional, apenas quando o utilizador grava um som) e histórico de gravações (associado à conta do utilizador). Todos os dados pessoais são armazenados de forma segura, encriptados em repouso e em trânsito, e nunca são partilhados com terceiros sem consentimento explícito.

Anonimização:

Para fins de investigação e treino de modelos, os dados são anonimizados, removendo informações identificáveis dos utilizadores. As coordenadas GPS são arredondadas para uma precisão de 1 km para proteger localizações exatas.

9.2. Uso Responsável de IA

Transparência:

Somos transparentes sobre as capacidades e limitações dos nossos modelos. Comunicamos claramente aos utilizadores que as previsões são probabilísticas e podem conter erros. Fornecemos níveis de confiança para cada previsão, permitindo aos utilizadores avaliar a fiabilidade. Disponibilizamos documentação sobre como os modelos funcionam e quais dados foram utilizados no treino.

Explicabilidade:

Implementamos técnicas de Explainable AI (XAI) para fornecer explicações sobre as previsões. Utilizamos Grad-CAM (Gradient-weighted Class Activation Mapping) para visualizar quais regiões do espectrograma foram mais importantes para a previsão. Fornecemos comparações com espécies similares para ajudar os utilizadores a compreender possíveis confusões.

Impacto Social:

O **BioLingo** foi desenhado para ter um impacto social positivo, contribuindo para a conservação da biodiversidade angolana, promovendo a educação ambiental e capacitando comunidades locais através de ciência cidadã. Estamos comprometidos em garantir que a tecnologia seja acessível e benéfica para todos os angolanos.

10. Conclusão

A documentação do projeto de aprendizado de máquina do BioLingo apresenta uma abordagem robusta, escalável e cientificamente fundamentada para a classificação automática de sons de animais. Através da combinação de transfer learning, arquiteturas de deep learning state-of-the-art e um pipeline de dados bem estruturado, o Grupo nº 14 está confiante em alcançar os objetivos de desempenho estabelecidos para o MVP.

A estratégia de implantação baseada em contêineres, microsserviços e infraestrutura de nuvem garante que o sistema será escalável, resiliente e capaz de servir milhares de utilizadores em Angola. O compromisso com o aprendizado contínuo, através de ciclos de feedback e retreino periódico, assegura que os modelos melhorarão continuamente ao longo do tempo.

O BioLingo representa não apenas um avanço tecnológico, mas também uma contribuição significativa para a conservação da biodiversidade, educação ambiental e capacitação científica em Angola. O Grupo nº 14 está empenhado em entregar um projeto de excelência que tenha impacto real e duradouro no nosso país.

Referências Técnicas

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-778.

[2] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

[3] Grill, T., & Schlüter, J. (2017). Two convolutional neural networks for bird detection in audio signals. *25th European Signal Processing Conference (EUSIPCO)*, 1764-1768.

[4] Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021). BirdNET: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61, 101236.

[5] Stowell, D., & Plumbley, M. D. (2014). Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2, e488.

[6] Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8024-8035.

[7] McFee, B., Raffel, C., Liang, D., et al. (2015). librosa: Audio and music signal analysis in python. *Proceedings of the 14th Python in Science Conference*, 18-25.

[8] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623-2631.