



UNIVERSIDADE EDUARDO MONDLANE

Faculdade de Engenharia

Departamento de Electrotécnica

Licenciatura em Engenharia Informática

INTELIGÊNCIA ARTIFICIAL II

Desenvolvimento de uma IA Tradutora da Língua de Sinais de Moçambique

Discentes:

Elihud, Clementina Uwimana

Florêncio, Nuno Fonseca

Matusse, Victor

Simão Júnior, Belarmino

Docente:

Eng°. Rúben Manhiça

Maputo, Março de 2024

Índice

Índice de Figuras	3
Introdução	4
Palavras-chave:	4
Objectivos	5
Objectivo geral.....	5
Objectivos específicos	5
Metodologias.....	6
1. Revisão da Literatura	7
1.1. Processo de Ciência de Dados.....	7
1.1.1. Formulação do Problema	7
1.1.2. Colecta de Dados	7
1.1.3. Pré-processamento dos Dados	7
1.1.4. Análise Exploratória dos Dados.....	7
1.1.5. Modelagem dos Dados.....	8
1.1.6. Interpretação e Comunicação de Resultados	8
1.2. Modelagem de Inteligência Artificial em Python	8
1.2.1. Biblioteca MediaPipe.....	8
1.2.2. Biblioteca Tensorflow.....	9
2. Etapas de Desenvolvimento do Modelo	10
2.1. Obtenção de dados	10
2.2. Preparação dos Dados	11
2.3. Exploração dos dados.....	12
2.4. Modelagem.....	13

2.4.1.	Arquitetura da Rede Neural	13
2.4.2.	Treinamento da Rede Neural	14
2.5.	Avaliação do Modelo	15
2.5.1.	Análise da Perda e Precisão	16
2.5.2.	Predições e Matriz de Confusão	17
3.	Resultados	19
	Conclusão.....	21
	Referências bibliográficas.....	22

Índice de Figuras

Figure 1. Detalhes dos Handmarks	9
Figure 2. Directório de armazenamento dos dados colectados	10
Figure 3. Captura de um frame com os pontos das mãos detectados pelo modelo MediaPipe	10
Figure 4. Mapeamento dos pontos gerados pelo modelo Holistic	11
Figure 5. Trecho de código que separa o conjunto de dados	12
Figure 6. Visualização de algumas amostras do dataset	13
Figure 7. Arquitectura da Rede Neural	14
Figure 8. Trecho de código de compilação e treinamento da Rede Neural	15
Figure 9. Gráfico da Perda nos dados de treino e validação ao longo das épocas	16
Figure 10. Gráfico da Precisão nos dados de treino e validação ao longo das épocas	16
Figure 11. Matriz de confusão nos dados de treino	17
Figure 12. Matriz de confusão nos dados de teste	17
Figure 13. Imagem de Resultado #1	19
Figure 14. Imagem de Resultado #2	19
Figure 15. Imagem de Resultado #3	20

Introdução

Um dos factores que cria uma barreira entre os humanos é a comunicação. Mas há muito menos comunicação entre aqueles que usam a linguagem gestual como principal meio de comunicação. Para diminuir a lacuna de comunicação entre utilizadores de língua gestual e não utilizadores da mesma, o grupo decidiu desenvolver uma inteligência artificial capaz de traduzir a linguagem de sinais de Moçambique para o português a fim de contribuir para a inclusão e a acessibilidade das pessoas surdas na sociedade.

Palavras-chave:

Inteligência Artificial, Língua de Sinais de Moçambique, Redes Neurais

Objectivos

Objectivo geral

Desenvolver e implementar uma inteligência artificial capaz de traduzir o alfabeto da linguagem de sinais de Moçambique para o português

Objectivos específicos

- Criar um conjunto de dados diversificado de gestos do alfabeto Língua de sinais de Moçambique para treinar o modelo de IA;
- Seguir o processo de ciência de dados para criar o modelo;
- Avaliar a precisão do modelo criado;
- Testar o modelo para novos dados.

Metodologias

O presente trabalho consistiu na pesquisa bibliográfica para colecta de informações acerca do projecto. Se fez uma colecta de dados do nosso dataset para o processo de treinamento de nosso modelo através da captura de fotos sinalizando o alfabeto na língua de sinais, no processo de colecta de dados ocorre também o processamento de dados capturando as marcas das mãos e organizou-se os dados em directórios e denominou-se os *labels* de cada directório. Fez-se o treinamento do nosso modelo. Por fim, fez-se uma validação do modelo, de modo a apurar a precisão e a eficácia do modelo.

1. Revisão da Literatura

1.1. Processo de Ciência de Dados

A ciência de dados é uma área que utiliza técnicas estatísticas e computacionais para analisar, compreender e extrair insights valiosos de grandes volumes de dados. Ela combina diversos campos, como estatística, matemática, programação e conhecimento de negócios, para resolver problemas complexos e tomar decisões baseadas em evidências.

1.1.1. Formulação do Problema

Antes de iniciar qualquer projecto de ciência de dados, é crucial determinar claramente o problema a ser resolvido. Isso envolve entender as necessidades e os objectivos do negócio, identificar as perguntas-chave a serem respondidas e definir as métricas de sucesso.

1.1.2. Colecta de Dados

Uma vez definido o problema, é necessário colectar os dados relevantes para análise. Esses dados podem estar disponíveis em diferentes formatos e fontes, como bancos de dados, arquivos CSV, APIs, páginas da web, entre outros. É importante garantir a qualidade e a integridade dos dados colectados.

1.1.3. Pré-processamento dos Dados

Antes de analisar os dados, é preciso realizar o pré-processamento, que inclui etapas como limpeza, transformação, integração e redução de dimensionalidade. Essas etapas visam garantir que os dados estejam prontos para a análise, livres de ruídos, inconsistências e redundâncias.

1.1.4. Análise Exploratória dos Dados

A análise exploratória visa entender e explorar os dados por meio de técnicas estatísticas e visualização. Nessa etapa, são identificadas tendências, padrões, relações e insights iniciais que podem orientar as próximas etapas do processo.

1.1.5. Modelagem dos Dados

Com base na análise exploratória, é possível desenvolver modelos estatísticos e algoritmos de *Machine Learning* para extrair insights mais profundos e fazer previsões. Esses modelos podem ser aplicados a conjuntos de dados de treinamento e validação para testar sua eficácia.

1.1.6. Interpretação e Comunicação de Resultados

A última etapa envolve interpretar os resultados da análise e comunicá-los de forma clara e acessível para as partes interessadas. Isso inclui a criação de relatórios, visualizações, *dashboards* e apresentações que ajudem na tomada de decisões e na implementação de acções.

1.2. Modelagem de Inteligência Artificial em Python

A modelagem de inteligência artificial em Python é o uso de bibliotecas e frameworks poderosos para desenvolver e implementar algoritmos de aprendizado de máquina.

1.2.1. Biblioteca MediaPipe

A biblioteca MediaPipe foi desenvolvido pela Google e aplica vários modelos de aprendizagem de máquina que trabalham juntos para identificar e rastrear mãos, rosto, o corpo humano e objectos do quotidiano em vídeos e imagens. Esta biblioteca oferece várias soluções que podem ser utilizadas para reconhecimento e rastreamento de partes do corpo humano e objectos em várias plataformas como web, mobile e computadores(MEDIAPIPE, 2022).

A documentação da solução Hands usada nos módulos responsáveis por detectar as configurações de mãos, movimento, localização e orientação da palma da mão ajudou a identificar uma forma de capturar as posições dos hand landmarks (pontos que o framework projecta em locais específicos da mão em cada frame do vídeo). Cada landmark tem um identificador como mostrado na Figura 1.



Figure 1. Detalhes dos Handmarks

1.2.2. Biblioteca Tensorflow

Criado pela equipe do Google Brain, o Tensorflow é um framework de código aberto para machine learning, focado em computação numérica de alto desempenho, especialmente projectado para permitir a criação e treinamento de redes neurais e modelos de aprendizado de máquina. É amplamente utilizada para a construção e treinamento de modelos de aprendizado devido à sua flexibilidade e capacidade de processamento em larga escala(site EITCA Academy).

No contexto do presente projecto, a biblioteca Tensorflow é usada principalmente para carregar um modelo pré-treinado (responsável por detectar e rastrear pontos de referência manuais na imagem.), processar quadros das imagens do conjunto de dados e fazer previsões com base nos dados de entrada.

2. Etapas de Desenvolvimento do Modelo

Para desenvolver e treinar o modelo de detecção do alfabeto em língua de sinais, seguimos as etapas do processo de ciência de dados (*data science process*). Essas etapas incluem a obtenção dos dados dos sinais, a preparação dos dados, a exploração dos dados, a modelagem e avaliação do modelo.

2.1. Obtenção de dados

Os dados necessários para o treinamento do modelo consistem em pontos das mãos. Para isso, optou-se pela captura automatizada de pontos das mãos através de um programa em Python que utiliza um modelo pré-treinado do MediaPipe, disponibilizado pela Google. Esse modelo permite a percepção simultânea, em tempo real, de pose humana, marcações faciais e rastreamento das mãos.

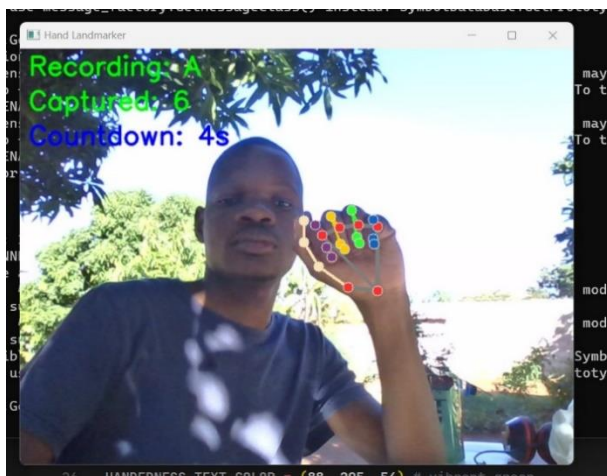


Figure 3. Captura de um frame com os pontos das mãos detectados pelo modelo MediaPipe

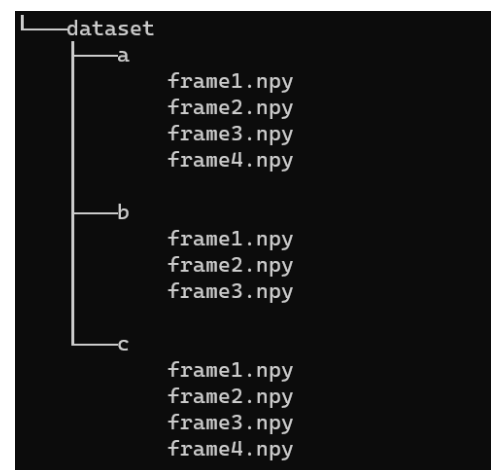


Figure 2. Directório de armazenamento dos dados colectados

Cada quadro capturado através da câmera passou pelo modelo **MediaPipe Holistic**, que detecta a presença e as coordenadas dos pontos nas mãos. Cada *frame* foi rotulado com o sinal equivalente e armazenado em directórios específicos.

2.2. Preparação dos Dados

Na etapa de preparação, as coordenadas dos pontos retornados pelo modelo Holistic são lidas do directório e convertidas para arrays do NumPy. O modelo Holistic retorna 21 pontos por mão, e cada ponto contém coordenadas em 3 dimensões (x, y e z), que indicam a localização do ponto da mão na tela.

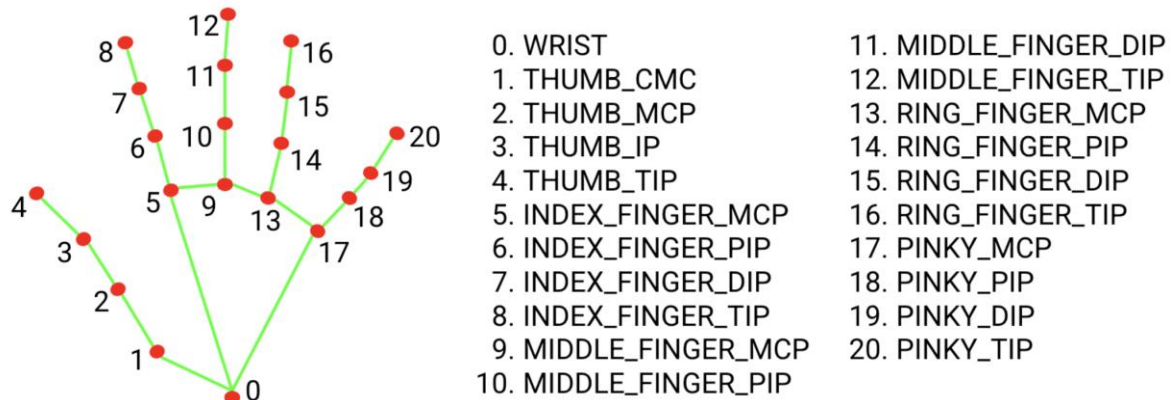
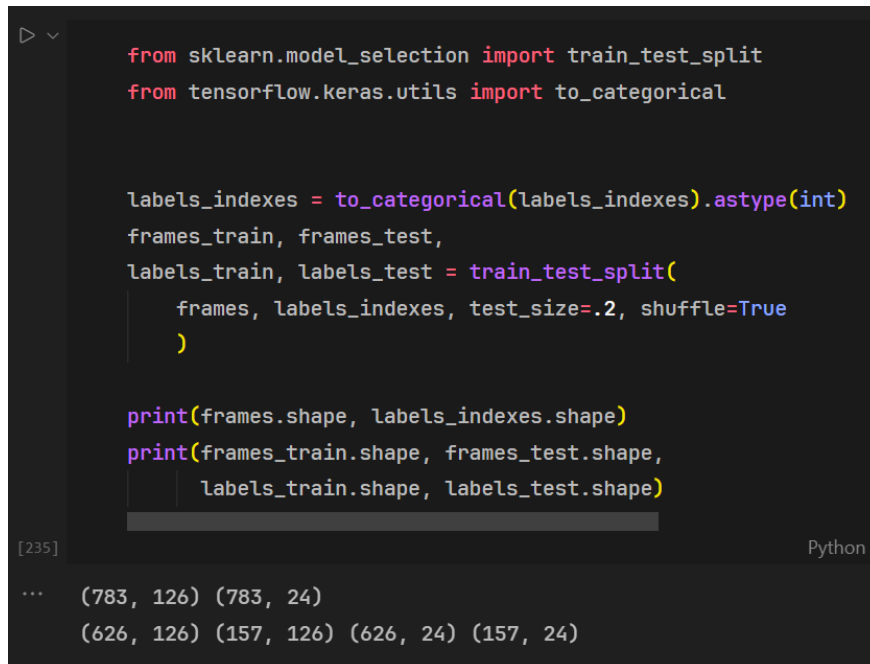


Figure 4. Mapeamento dos pontos gerados pelo modelo Holistic

Em seguida, as coordenadas são organizadas em arrays unidimensionais, totalizando 126 elementos em cada array (21 pontos * 3 coordenadas * 2 mãos). Esse formato será útil futuramente como entrada para o modelo. Arrays com valores nulos, que representam a ausência de mãos detectadas, foram descartados.



```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

labels_indexes = to_categorical(labels_indexes).astype(int)
frames_train, frames_test,
labels_train, labels_test = train_test_split(
    frames, labels_indexes, test_size=.2, shuffle=True
)

print(frames.shape, labels_indexes.shape)
print(frames_train.shape, frames_test.shape,
      labels_train.shape, labels_test.shape)
```

[235] Python

... (783, 126) (783, 24)
(626, 126) (157, 126) (626, 24) (157, 24)

Figure 5. Trecho de código que separa o conjunto de dados

No final da etapa, o conjunto de dados (783 capturas) foi dividido em conjunto de treino (80% dos dados) e de teste usando o módulo sklearn-python. Os rótulos (que correspondem ao alfabeto) são convertidos em rótulos categóricos.

2.3. Exploração dos dados

Nesta etapa de exploração foi feita a visualização de dados, que permite entender e analisar a distribuição e características dos gestos capturados. Para isso, alguns gestos específicos foram seleccionados para fins de visualização.

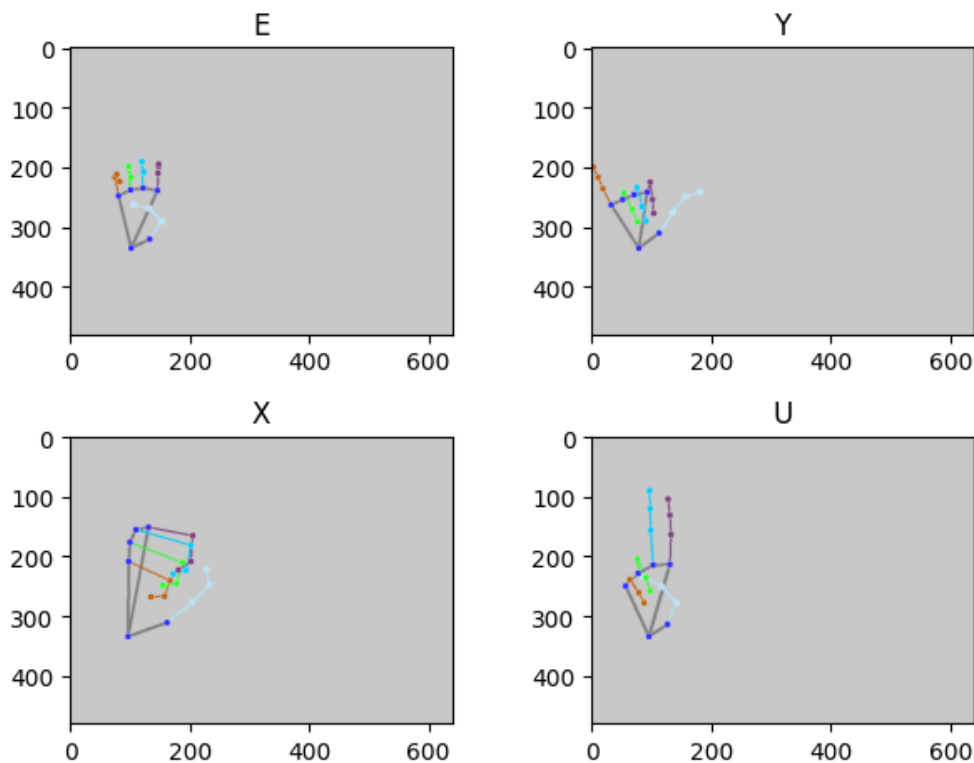


Figure 6. Visualização de algumas amostras do dataset

As bibliotecas usadas para a visualização foram o módulo de desenho do MediaPipe e o matplotlib. A visualização assegura que os dados estão prontos e adequadamente formatados para serem utilizados no treinamento do modelo de detecção de alfabetos em língua de sinais.

2.4. Modelagem

Para alcançar o objectivo definido, foi decidido o uso de uma Rede Neural Artificial (RNA), devido à sua capacidade de aprender e generalizar padrões complexos a partir dos dados de entrada.

2.4.1. Arquitectura da Rede Neural

A arquitectura da rede neural foi projectada para lidar com os dados de pontos das mãos e realizar a classificação correcta dos gestos do alfabeto.

Layer (type)	Output Shape	Param #
dense_20 (Dense)	(None, 128)	16,256
dropout_10 (Dropout)	(None, 128)	0
dense_21 (Dense)	(None, 64)	8,256
dropout_11 (Dropout)	(None, 64)	0
dense_22 (Dense)	(None, 16)	1,040
dense_23 (Dense)	(None, 24)	408

Total params: 25,960 (101.41 KB)

Trainable params: 25,960 (101.41 KB)

Non-trainable params: 0 (0.00 B)

Figure 7. Arquitetura da Rede Neural

Camada de Entrada: Consiste em 126 neurónios artificiais, correspondentes aos 126 elementos dos arrays de entrada (21 pontos por mão * 3 coordenadas por ponto * 2 mãos).

Camadas Ocultas: Diversas camadas densas de neurónios foram adicionadas para aumentar a capacidade da rede de capturar padrões complexos. O número exacto de camadas e neurónios em cada camada foi determinado através de experimentação e ajustes.

Camada de Saída: Consiste em 24 neurónios artificiais, representando as 24 letras do alfabeto (excepto as letras "J" e "Z", que exigem sequências de gestos e não podem ser representadas em um único frame). A camada de saída utiliza a função de activação Softmax, que distribui a saída em probabilidades, indicando o nível de confiança das predições do modelo.

2.4.2. Treinamento da Rede Neural

Para treinar a rede neural, o modelo foi compilado com o optimizador Adam, uma escolha popular devido à sua eficiência e capacidade de ajuste adaptativo da taxa de aprendizado. A função de perda utilizada foi a entropia cruzada categórica (categorical_crossentropy), adequada para tarefas

de classificação multiclass. Além disso, a métrica de avaliação escolhida foi a precisão (accuracy), permitindo monitorar o desempenho do modelo durante o treinamento.

```
# Compilação do modelo
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Treino do modelo
history = model.fit(
    frames_train, labels_train,
    validation_data=(frames_test, labels_test),
    epochs=100,
    callbacks=[earlystopping])
```

Figure 8. Trecho de código de compilação e treinamento da Rede Neural

O treinamento do modelo foi conduzido por 100 épocas (epochs=100), garantindo que o modelo tivesse tempo suficiente para aprender os padrões nos dados de treinamento. Para evitar o overfitting e melhorar a generalização, foi utilizado o callback de EarlyStopping, monitorando a perda na validação. Esse callback foi configurado para interromper o treinamento se a perda na validação não melhorasse após épocas consecutivas e para restaurar os melhores pesos do modelo observados durante o treinamento.

2.5. Avaliação do Modelo

A avaliação do modelo é consiste em entender seu desempenho e identificar áreas de melhoria. O processo de avaliação inclui a análise da perda e da precisão durante o treinamento, bem como a geração e visualização de uma matriz de confusão para verificar a eficácia das predições.

2.5.1. Análise da Perda e Precisão

Os históricos de perda e precisão durante o treinamento e validação foram plotados. Isso ajuda a visualizar como o modelo se comportou ao longo das épocas, permitindo identificar sinais de overfitting ou underfitting.

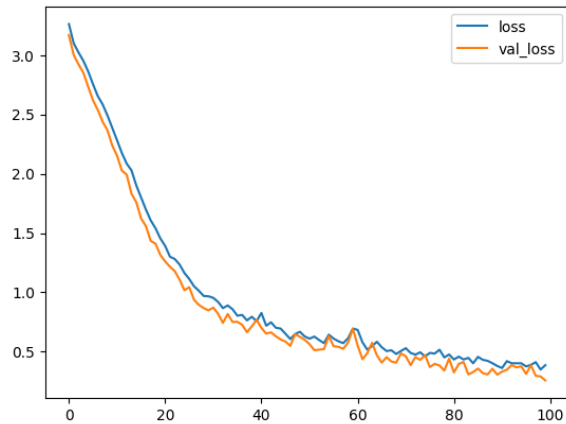


Figure 9. Gráfico da Perda nos dados de treino e validação ao longo das épocas

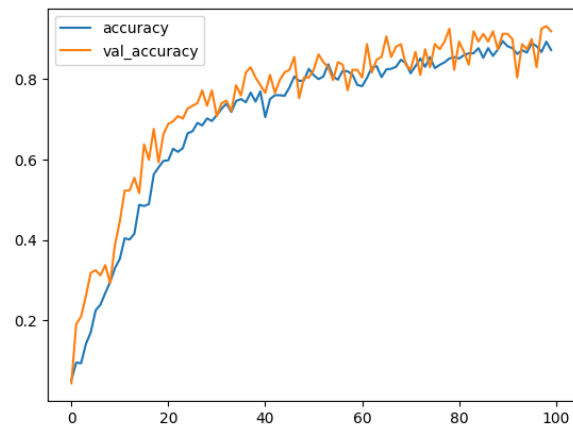


Figure 10. Gráfico da Precisão nos dados de treino e validação ao longo das épocas

O gráfico de perda (Figura 9) mostra a função de perda do modelo tanto nos dados de treino quanto nos dados de validação ao longo das 100 épocas de treinamento. Observa-se que:

- **Redução Contínua da Perda:** A perda diminui de forma consistente tanto nos dados de treino quanto nos de validação, indicando que o modelo está aprendendo os padrões presentes nos dados.
- **Convergência:** A perda nos dados de validação acompanha de perto a perda nos dados de treino, o que sugere que o modelo não está sobre ajustando aos dados de treino.
- **Estabilização:** Aproximadamente após 60 épocas, a perda começa a estabilizar, com melhorias menores nas épocas subsequentes.

O gráfico de precisão (Figura 10) mostra a métrica de precisão do modelo tanto nos dados de treino quanto nos dados de validação ao longo das 100 épocas de treinamento. Observa-se que:

- **Aumento Contínuo da Precisão:** A precisão aumenta consistentemente em ambas as curvas, indicando que o modelo está melhorando sua capacidade de classificação.
- **Alta Precisão Final:** A precisão nos dados de validação aproxima-se da precisão nos dados de treino, sugerindo que o modelo está generalizando bem para dados não vistos.

- Convergência e Estabilização: Por volta das 60 épocas, a precisão começa a estabilizar, com o modelo atingindo uma precisão próxima a 90%, o que é um excelente resultado para este tipo de tarefa.

2.5.2. Predições e Matriz de Confusão

Após o treinamento, o modelo foi utilizado para fazer predições nos dados de treinamento. As predições e os rótulos reais foram convertidos para índices de classe, que representam as letras do alfabeto.

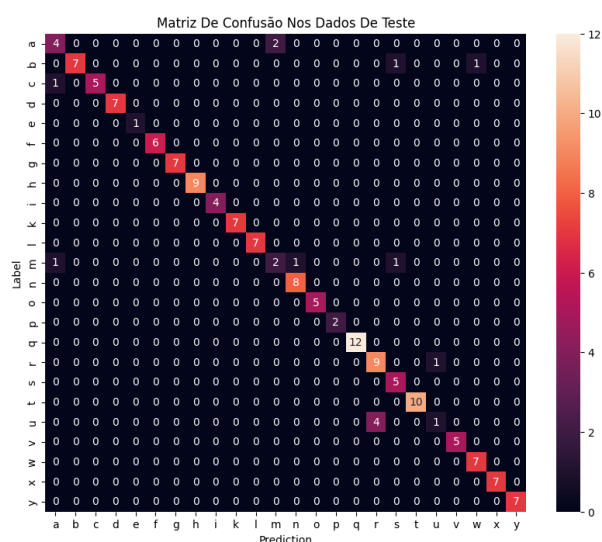


Figure 12. Matriz de confusão nos dados de teste

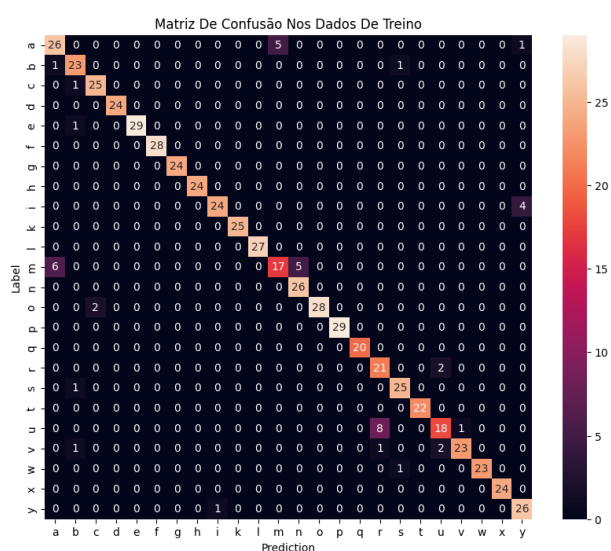


Figure 11. Matriz de confusão nos dados de treino

Cada célula na matriz de confusão representa a quantidade de vezes que uma letra específica foi prevista como outra letra. As células na diagonal principal mostram as previsões correctas, onde a letra prevista coincide com a letra real.

A Figura 11 apresenta a matriz de confusão nos dados de treino e a Figura 12 nos dados de teste. Na primeira matriz, observa-se que:

- A maioria das letras é correctamente classificada, como indicado pelas altas contagens na diagonal principal (ex. 'a' como 'a' 26 vezes, 'b' como 'b' 23 vezes, etc.).
- Há algumas confusões, mas são relativamente pequenas. Por exemplo, 'm' foi confundida como 'n' em 3 ocasiões e vice-versa.

Na primeira matriz, observa-se também que a maioria das letras é correctamente classificada, apesar do menor volume de dados.

A confusão entre as letras 'M' e 'A' é destacada em ambas as matrizes de confusão. Essa confusão é explicada pelo fato de que os gestos para estas letras na linguagem de sinais de Moçambique são bastante semelhantes, levando o modelo a ter dificuldade em diferenciá-las.

3. Resultados

Terminado o processo de treinamento e teste, podemos observar que o modelo faz a identificação do alfabeto com um nível alto de precisão e eficácia. Porém constatou-se que para sinais que usam um uso combinado de símbolos o nosso modelo não estava preparado para identificar pois ele baseia-se em captura de fotos, temos como um exemplo a letra z.

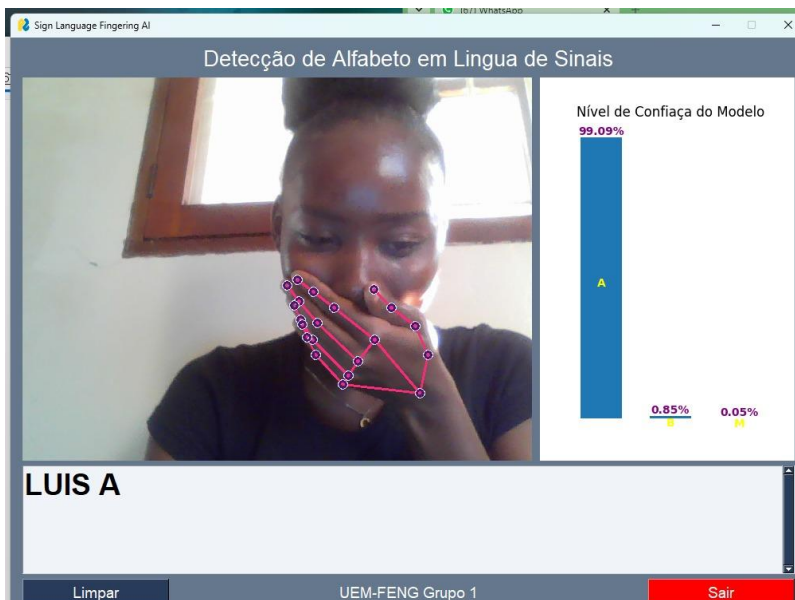


Figure 13. Imagem de Resultado #1

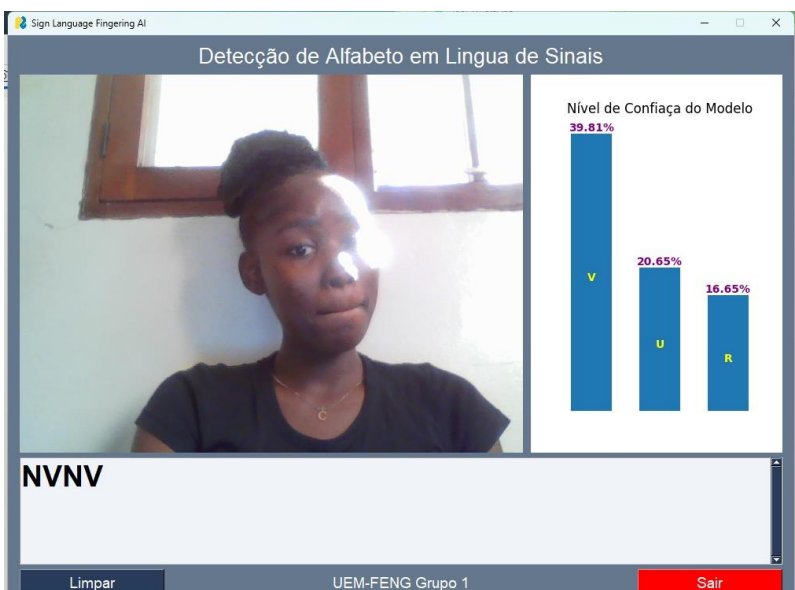


Figure 14. Imagem de Resultado #2

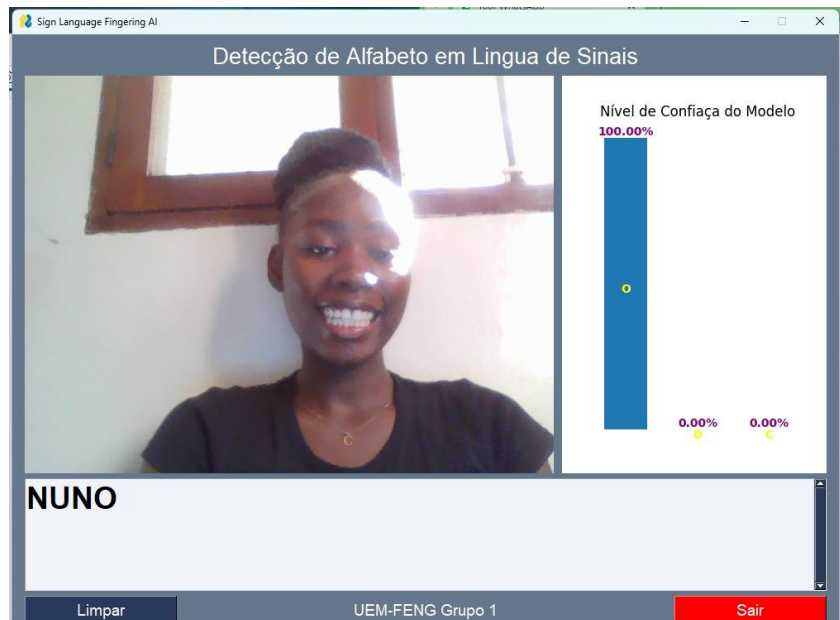


Figure 15. Imagem de Resultado #3

Conclusão

Este projecto desenvolveu uma inteligência artificial para traduzir o alfabeto da linguagem de sinais de Moçambique para o português, promovendo a inclusão das pessoas surdas na sociedade. Utilizando o modelo MediaPipe Holistic para capturar pontos das mãos, organizamos os dados em arrays unidimensionais e treinamos uma rede neural com camadas densas e função de activação Softmax. O modelo foi avaliado através de gráficos de perda e precisão e uma matriz de confusão, demonstrando alta precisão e boa generalização para dados não vistos, sem sinais de *overfitting*. Os resultados mostram que o modelo pode traduzir sinais do alfabeto com alta precisão, representando um avanço significativo na inclusão digital e social das pessoas surdas em Moçambique. Futuras melhorias podem expandir o modelo para reconhecer palavras e frases completas

Referências bibliográficas

1. Google AI. (2023). MediaPipe Hands: Um modelo de aprendizado de máquina para reconhecimento de gestos em tempo real. Google AI Edge. https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer?hl=pt-br
2. EITCA. (2023). Tag: TensorFlow. EITCA. <https://pt.eitca.org/tag/tensorflow/page/17/>
3. Awari. (2020). Seis passos do processo de ciência de dados. Awari. <https://awari.com.br/seis-passos-do-processo-de-ciencia-de-dados/>
4. Cielen, D., Meysman, A. D. B., & Ali, M. Introducing Data Science: Big Data, Machine Learning, and More, Using Python Tools.