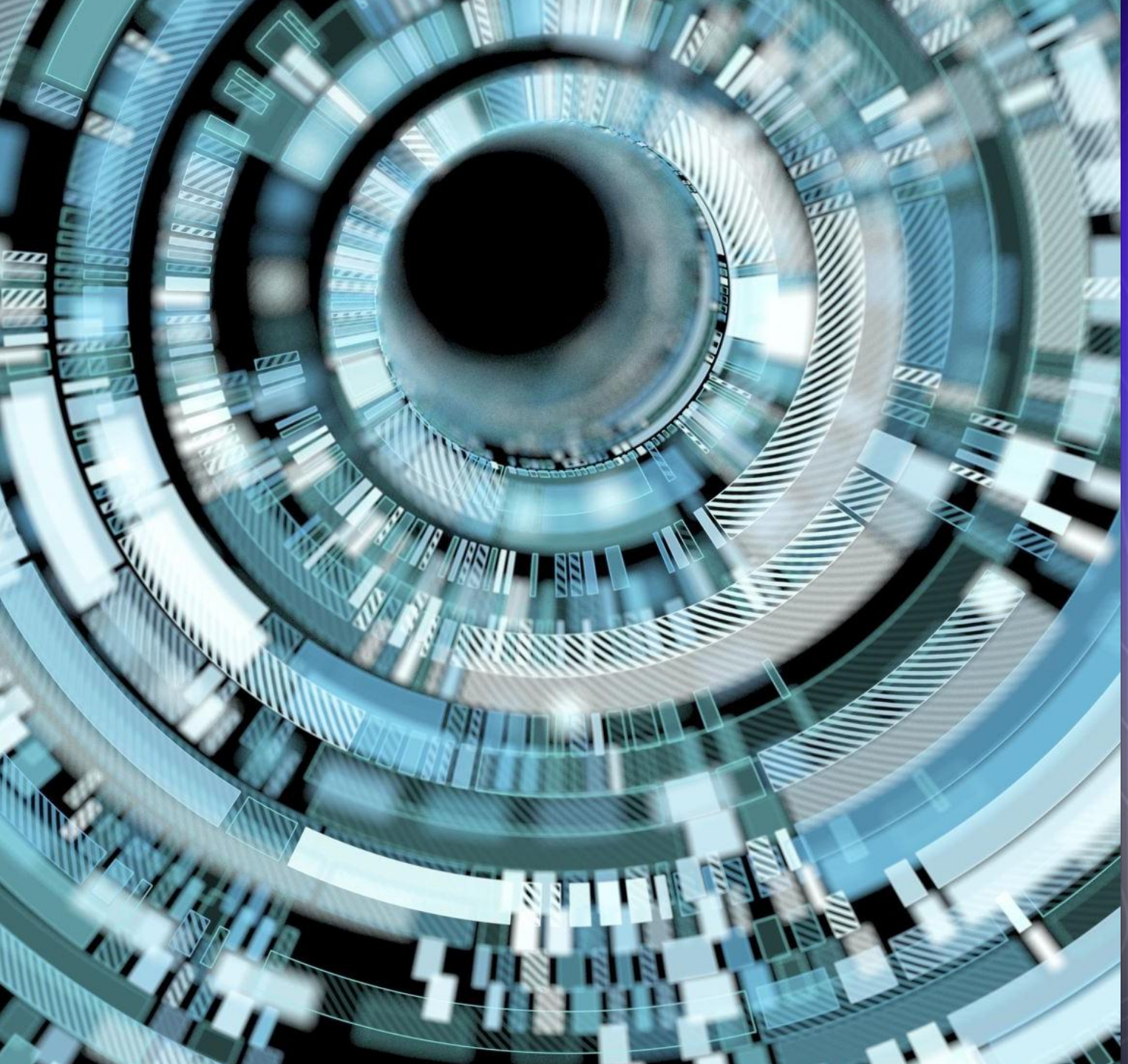


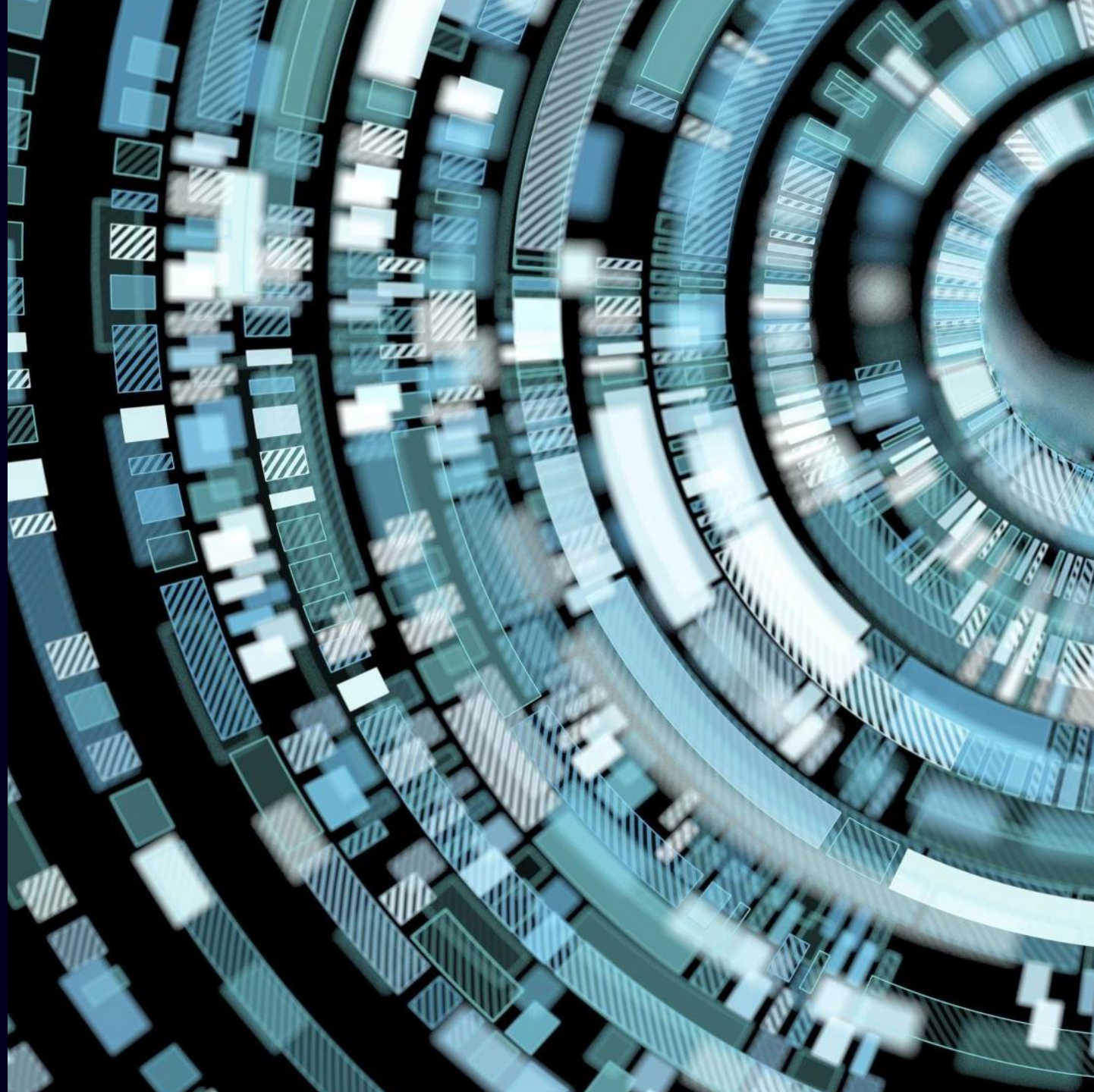
Visão Computacional para Interpretação de Libras

Centro Universitário FEI
Nityananda Vianna
Priscila Vazquez
Victor Ayres
2024

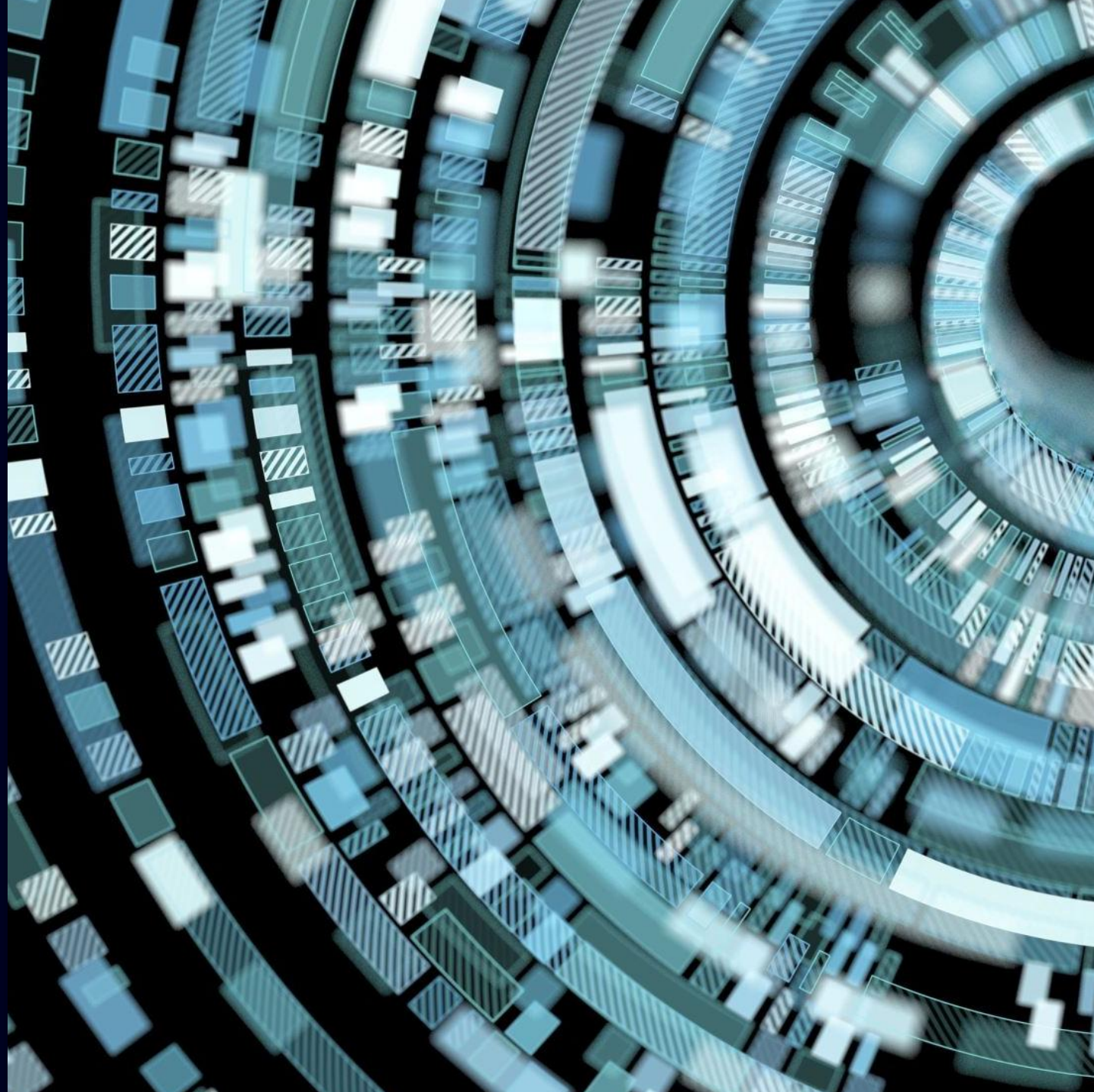
Resumo

- Este trabalho explora a aplicação de visão computacional para interpretar a Língua Brasileira de Sinais (Libras) utilizando a biblioteca MediaPipe e Redes Neurais Convolucionais 3D (CNN 3D).

- A acurácia atingida foi de 99,06%, demonstrando potencial para aplicações em tradução de Libras e acessibilidade.



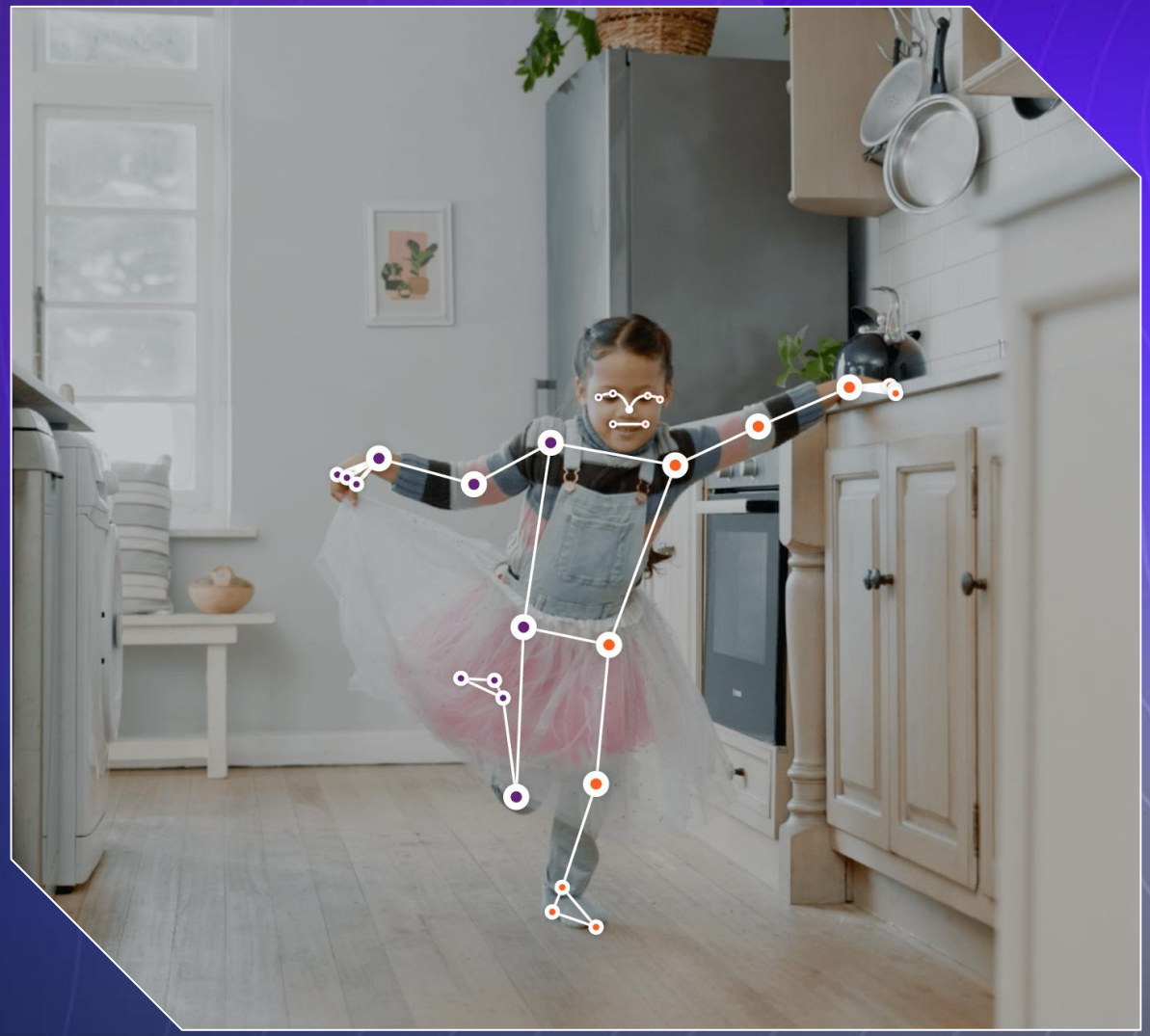
- INTRODUÇÃO
- PRÉ-PROCESSAMENTOS DOS DADOS
  - DATA AGUMENTATION
  - PADRONIZAO DOS VÍDEOS
  - SEPARAÇÃO DO DATASET
- ESTRUTURA DA REDE NEURAL
- RECONHECIMENTO EM TEMPO REAL
- APLICAÇÕES E POTENCIALIDADES



Introdução

O projeto utiliza bibliotecas como MediaPipe para reconhecimento de sinais em Libras. Uma Rede Neural Convolucional 3D (CNN 3D) foi treinada para identificar movimentos e reconhecer sinais com precisão.

Pré-processamento dos Dados



MIND-LIBRAS DATASET

EXTRAÇÃO DAS MASK COM MEDIAPIPE

TÉCNICAS PARA DATA AUGMENT

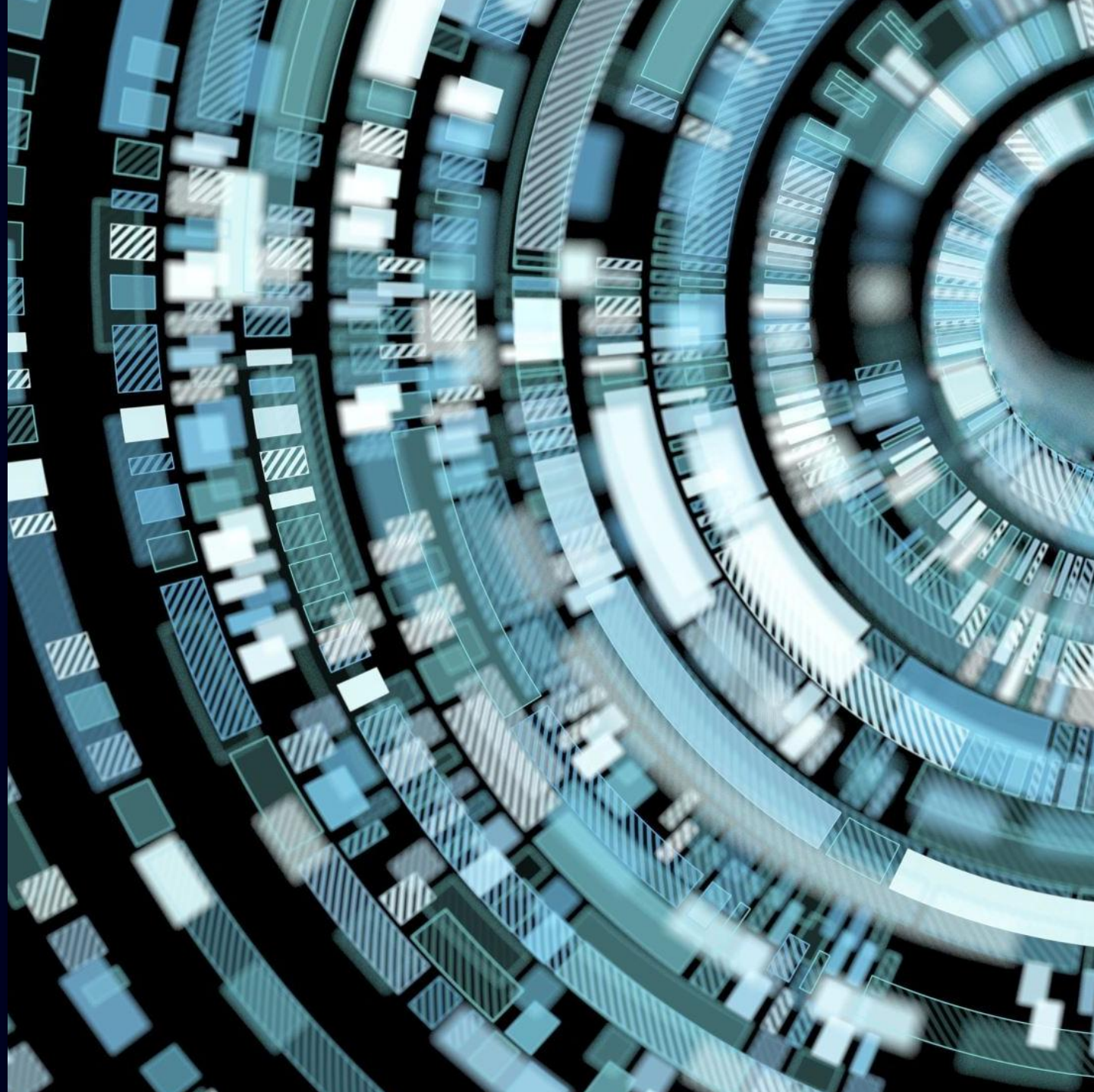
PADRONIZAÇÃO DOS VÍDEOS

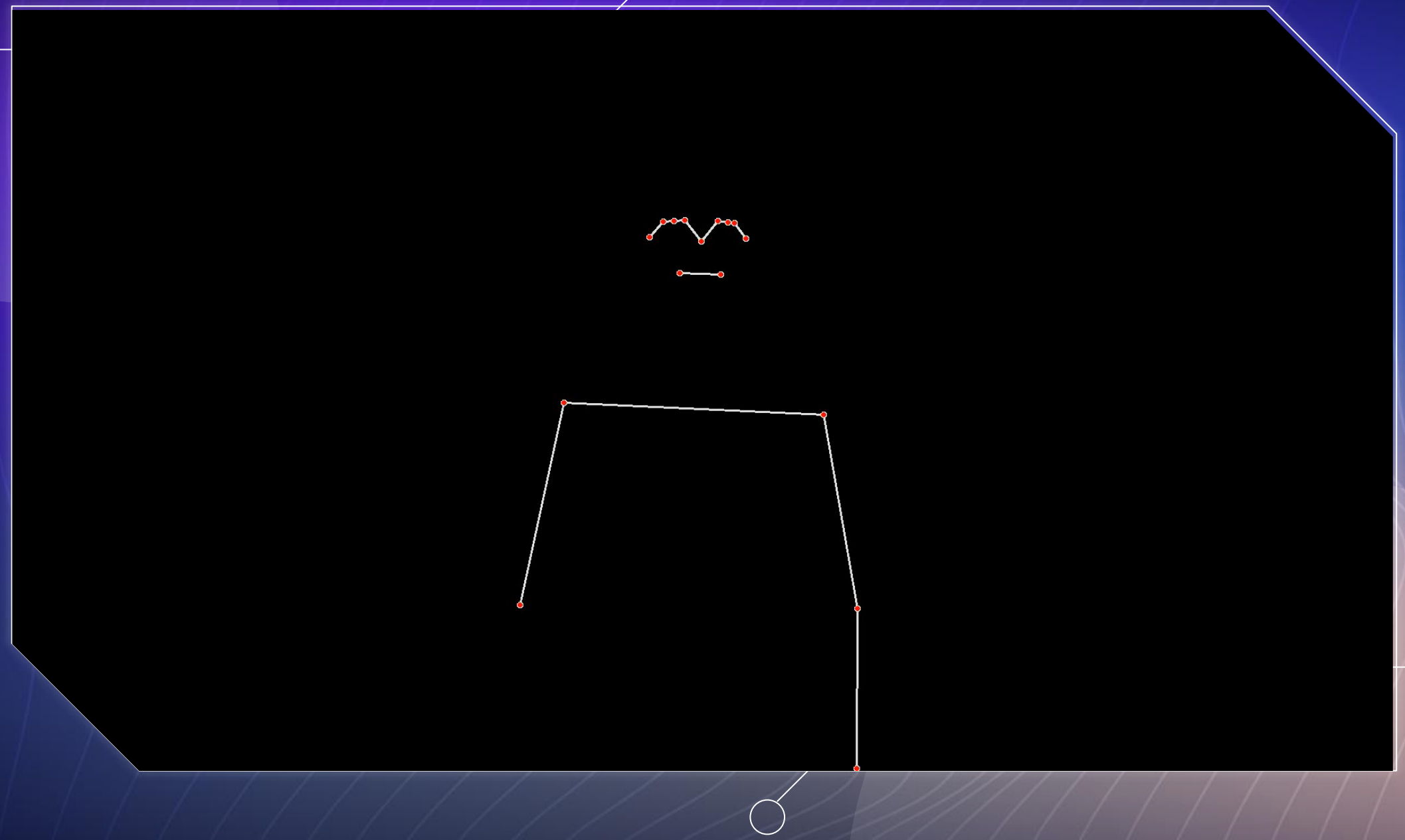
MASK COM MEDIAPIPE

- Framework desenvolvido pelo Google.
- Otimizado para rastreamento em tempo real.

- **Hand Tracking**: Identificação de posições das articulações da mão.
- **Pose Tracking**: Rastreio dos movimentos corporais para maior contexto.

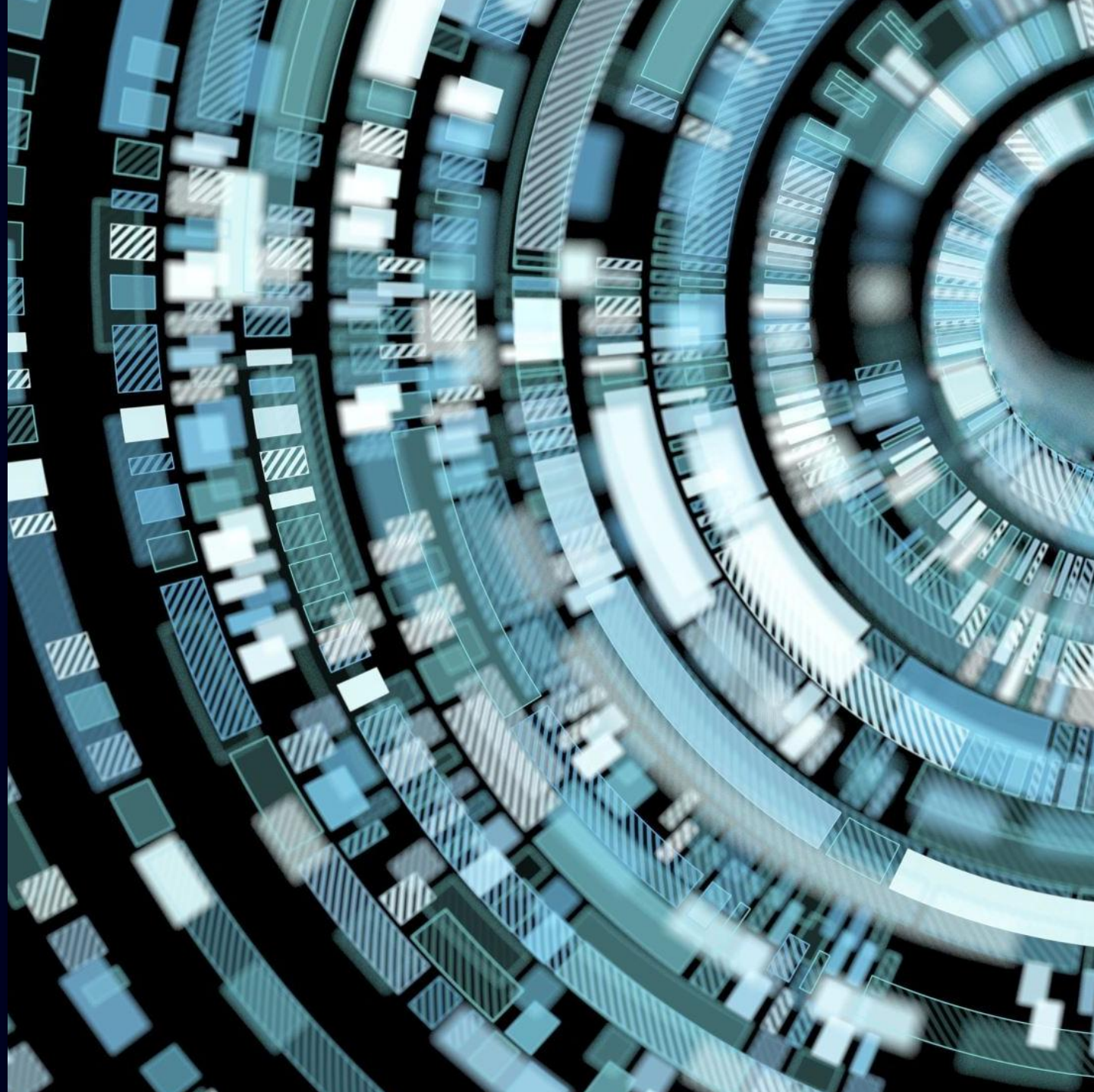
- Padronização dos sinais melhorou a distinção entre símbolos.





DATA AUGMENTATION

- **Técnicas Aplicadas**
  - **Zoom:** Ajustes de escala para simular variações de distância da câmera.
  - **Deslocamento Lateral:** Movimentação horizontal para simular diferentes posições no quadro.
- **Objetivo**
  - Aumentar a diversidade do dataset.
  - Tornar o modelo mais robusto a variações espaciais nos sinais.



```python
# Função para aplicar zoom no centro da imagem
def apply_zoom(image, zoom_factor=1.2):
    height, width = image.shape[:2]
    new_height, new_width = int(height / zoom_factor), int(width / zoom_factor)
    y1, x1 = (height - new_height) // 2, (width - new_width) // 2
    y2, x2 = y1 + new_height, x1 + new_width
    zoomed_image = cv.resize(image[y1:y2, x1:x2], (width, height))
    return zoomed_image
```

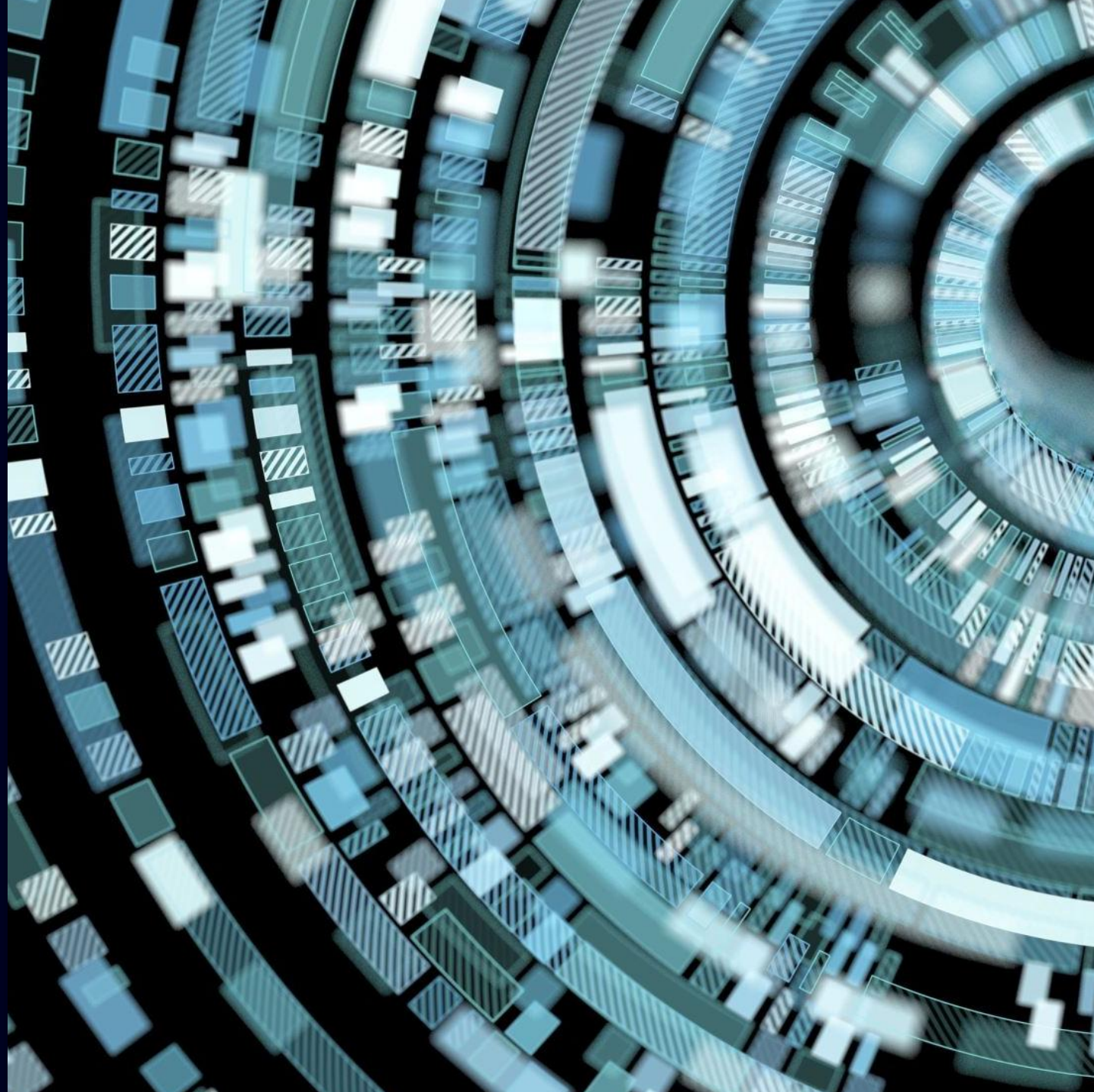
```python
# Função para deslocar a imagem para a esquerda ou direita
def shift_image(image, shift_pixels):
    height, width = image.shape[:2]
    M = np.float32([[1, 0, shift_pixels], [0, 1, 0]])
    shifted_image = cv.warpAffine(image, M, (width, height))
    return shifted_image

```

PADRONIZANDO OS VÍDEOS

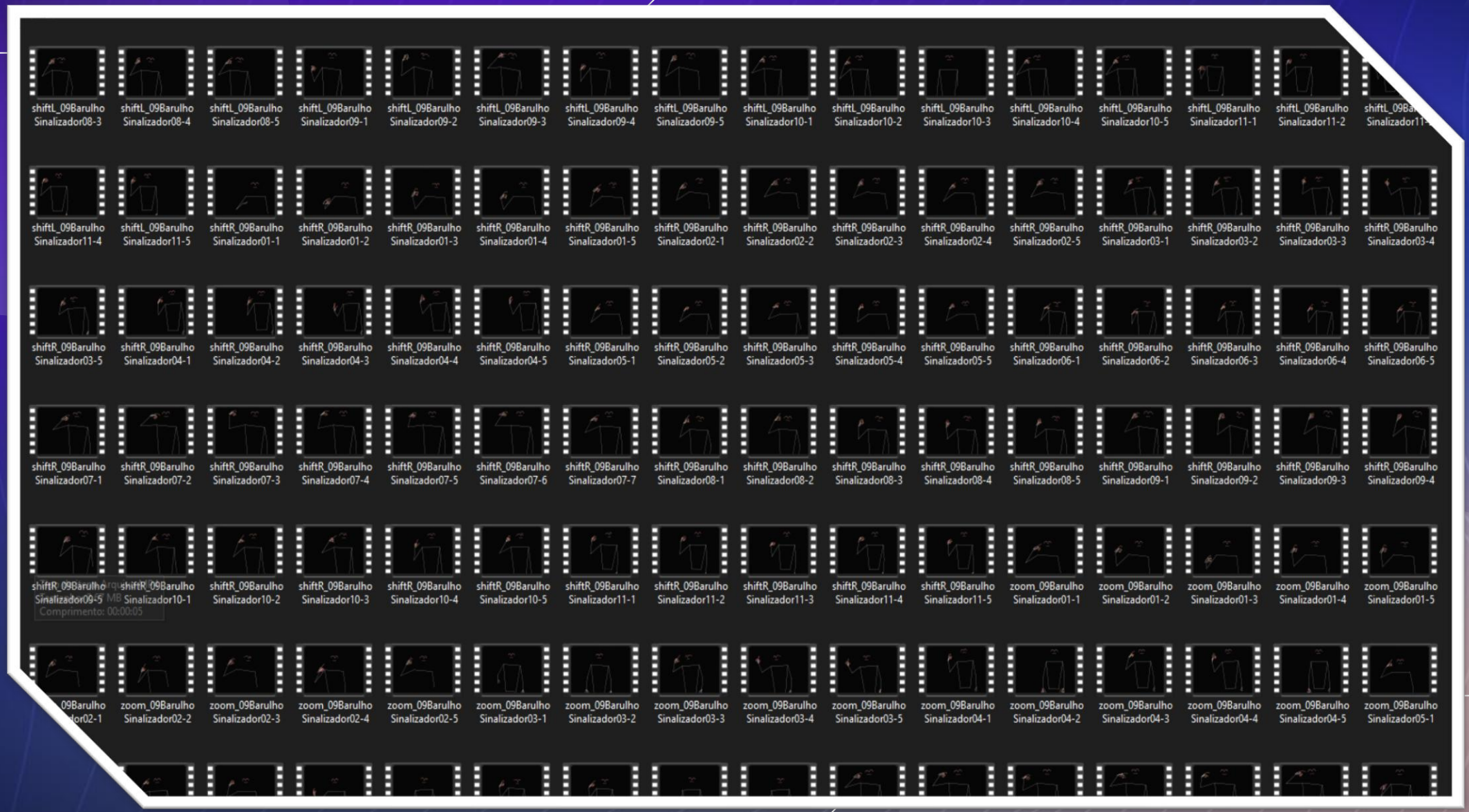
- **Uniformização dos Frames**
  - Todos os vídeos ajustados para a mesma quantidade de frames (60).
  - Aplicação de **padding** quando necessário para completar sequências menores.

- **Conversão para Escala de Cinza**
  - Simplificação dos dados para reduzir a complexidade computacional.

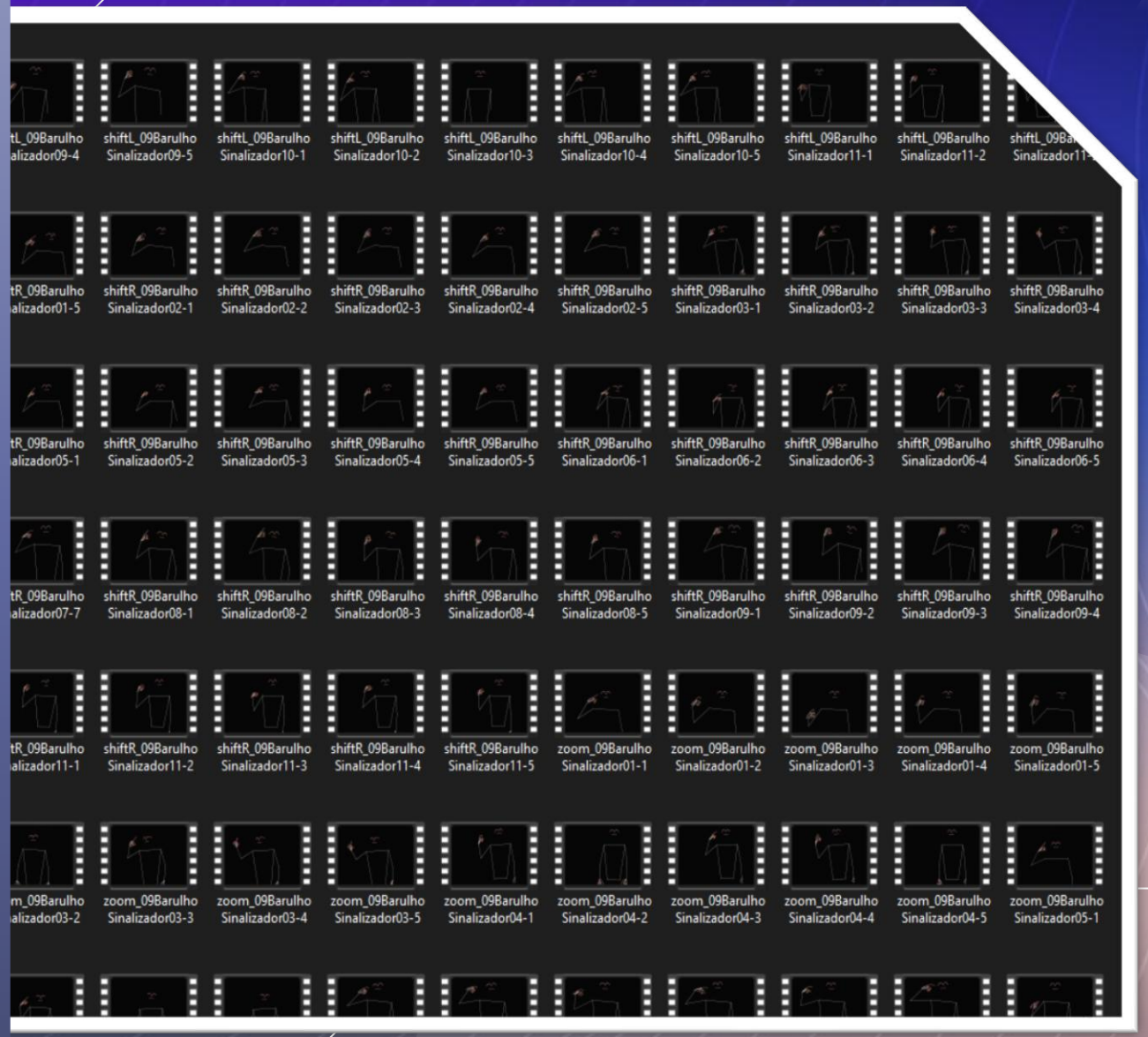
- **Redimensionamento**
  - Resolução ajustada para **64x64 pixels.**
  - Compatibilidade com a entrada da CNN 3D.



```python
# Função de processamento de vídeos
def process_video(video_path, target_frames=num_frames):
    cap = cv2.VideoCapture(video_path)
    frames = []
    success, frame = cap.read()

    # Loop para leitura dos frames
    while success:
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        resized_frame = cv2.resize(gray_frame, (largura, altura))
        frames.append(resized_frame / 255.0)  # Normalização entre 0 e 1
        success, frame = cap.read()

    cap.release()

    # Normalizaação do número de frames ( padding ou truncamento )
    if len(frames) > target_frames: # Truncamento
        indices = np.linspace(0, len(frames) - 1, target_frames).astype(int)
        frames = [frames[i] for i in indices]
    elif len(frames) < target_frames: # Padding
        padding = target_frames - len(frames)
        frames.extend([frames[-1]] * padding)

    return np.array(frames)
```

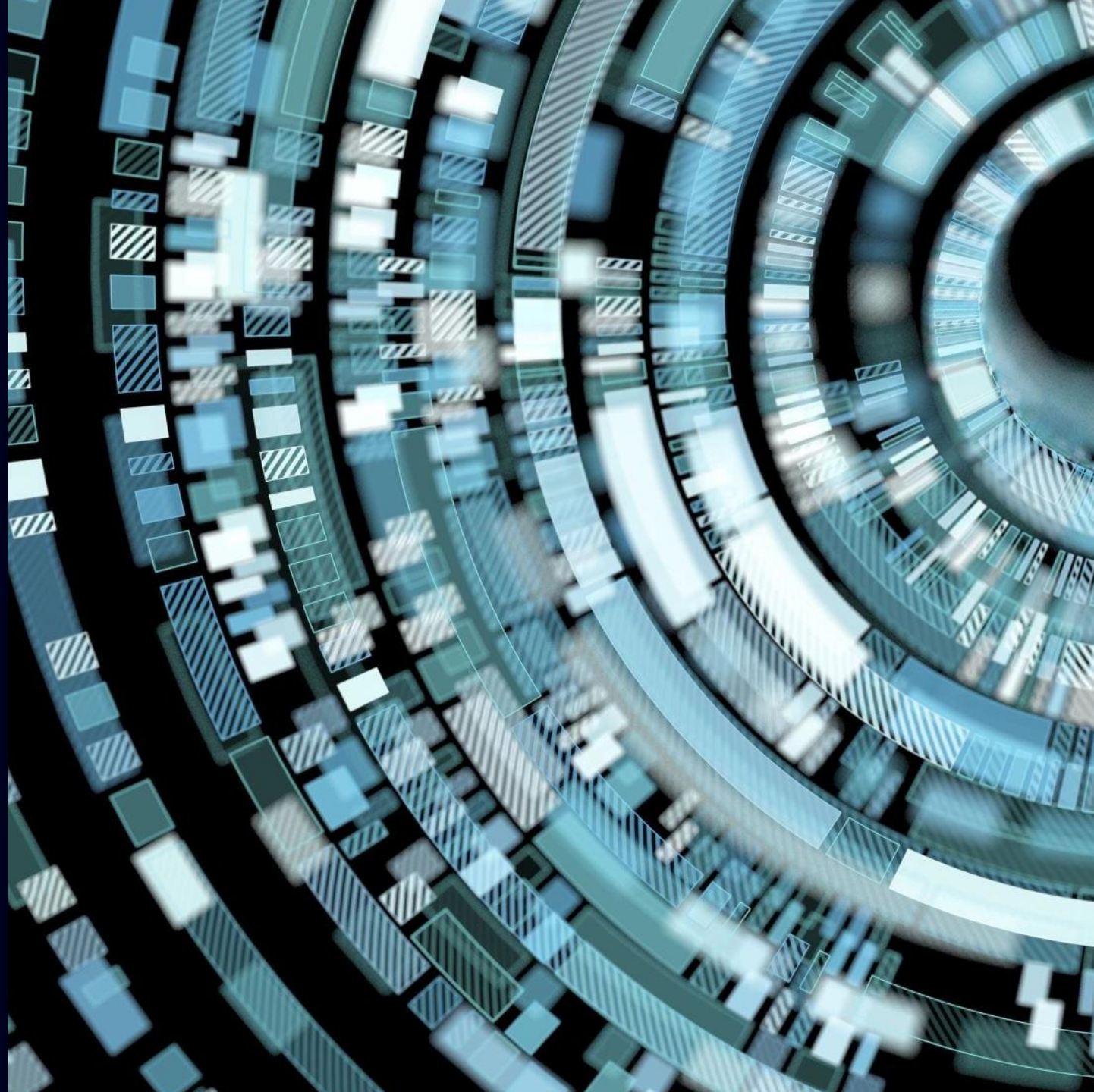

shiftL_09Barulho Sinalizador08-3
shiftL_09Barulho Sinalizador08-4
shiftL_09Barulho Sinalizador08-5
shiftL_09Barulho Sinalizador09-1
shiftL_09Barulho Sinalizador09-2
shiftL_09Barulho Sinalizador09-3
shiftL_09Barulho Sinalizador09-4
shiftL_09Barulho Sinalizador09-5
shiftL_09Barulho Sinalizador10-1
shiftL_09Barulho Sinalizador10-2
shiftL_09Barulho Sinalizador10-3
shiftL_09Barulho Sinalizador10-4
shiftL_09Barulho Sinalizador10-5
shiftL_09Barulho Sinalizador11-1
shiftL_09Barulho Sinalizador11-2
shiftL_09Barulho Sinalizador11-4
shiftL_09Barulho Sinalizador11-5
shiftR_09Barulho Sinalizador01-1
shiftR_09Barulho Sinalizador01-2
shiftR_09Barulho Sinalizador01-3
shiftR_09Barulho Sinalizador01-4
shiftR_09Barulho Sinalizador01-5
shiftR_09Barulho Sinalizador02-1
shiftR_09Barulho Sinalizador02-2
shiftR_09Barulho Sinalizador02-3
shiftR_09Barulho Sinalizador02-4
shiftR_09Barulho Sinalizador02-5
shiftR_09Barulho Sinalizador03-1
shiftR_09Barulho Sinalizador03-2
shiftR_09Barulho Sinalizador03-3
shiftR_09Barulho Sinalizador03-4
shiftR_09Barulho Sinalizador03-5
shiftR_09Barulho Sinalizador04-1
shiftR_09Barulho Sinalizador04-2
shiftR_09Barulho Sinalizador04-3
shiftR_09Barulho Sinalizador04-4
shiftR_09Barulho Sinalizador04-5
shiftR_09Barulho Sinalizador05-1
shiftR_09Barulho Sinalizador05-2
shiftR_09Barulho Sinalizador05-3
shiftR_09Barulho Sinalizador05-4
shiftR_09Barulho Sinalizador05-5
shiftR_09Barulho Sinalizador06-1
shiftR_09Barulho Sinalizador06-2
shiftR_09Barulho Sinalizador06-3
shiftR_09Barulho Sinalizador06-4
shiftR_09Barulho Sinalizador06-5
shiftR_09Barulho Sinalizador07-1
shiftR_09Barulho Sinalizador07-2
shiftR_09Barulho Sinalizador07-3
shiftR_09Barulho Sinalizador07-4
shiftR_09Barulho Sinalizador07-5
shiftR_09Barulho Sinalizador07-6
shiftR_09Barulho Sinalizador07-7
shiftR_09Barulho Sinalizador08-1
shiftR_09Barulho Sinalizador08-2
shiftR_09Barulho Sinalizador08-3
shiftR_09Barulho Sinalizador08-4
shiftR_09Barulho Sinalizador08-5
shiftR_09Barulho Sinalizador09-1
shiftR_09Barulho Sinalizador09-2
shiftR_09Barulho Sinalizador09-3
shiftR_09Barulho Sinalizador09-4
shiftR_09Barulho Sinalizador09-5
shiftR_09Barulho Sinalizador10-1
shiftR_09Barulho Sinalizador10-2
shiftR_09Barulho Sinalizador10-3
shiftR_09Barulho Sinalizador10-4
shiftR_09Barulho Sinalizador10-5
shiftR_09Barulho Sinalizador11-1
shiftR_09Barulho Sinalizador11-2
shiftR_09Barulho Sinalizador11-3
shiftR_09Barulho Sinalizador11-4
shiftR_09Barulho Sinalizador11-5
zoom_09Barulho Sinalizador01-1
zoom_09Barulho Sinalizador01-2
zoom_09Barulho Sinalizador01-3
zoom_09Barulho Sinalizador01-4
zoom_09Barulho Sinalizador01-5
zoom_09Barulho Sinalizador02-1
zoom_09Barulho Sinalizador02-2
zoom_09Barulho Sinalizador02-3
zoom_09Barulho Sinalizador02-4
zoom_09Barulho Sinalizador02-5
zoom_09Barulho Sinalizador03-1
zoom_09Barulho Sinalizador03-2
zoom_09Barulho Sinalizador03-3
zoom_09Barulho Sinalizador03-4
zoom_09Barulho Sinalizador03-5
zoom_09Barulho Sinalizador04-1
zoom_09Barulho Sinalizador04-2
zoom_09Barulho Sinalizador04-3
zoom_09Barulho Sinalizador04-4
zoom_09Barulho Sinalizador04-5
zoom_09Barulho Sinalizador05-1

- DATASET TEVE AUMENTO CONSIDERÁVEL DE ITENS. 60 VÍDEOS POR CLASSES PARA 240 POR CLASSE
- DREDUCO DO TAMANHO (PESO) DO DATA SET. DE 52GB PARA 8 GB
- MELHORIA NO TREINAMWENTO DA IA E MELHOR RECONHECIMENTO DOS SINAIS
- PRINCIPAL VANTAGEM: TREINAENTO ACAB TENDO O FOCO NO MOVIMENTO DOS BRAÇOS E MAOS



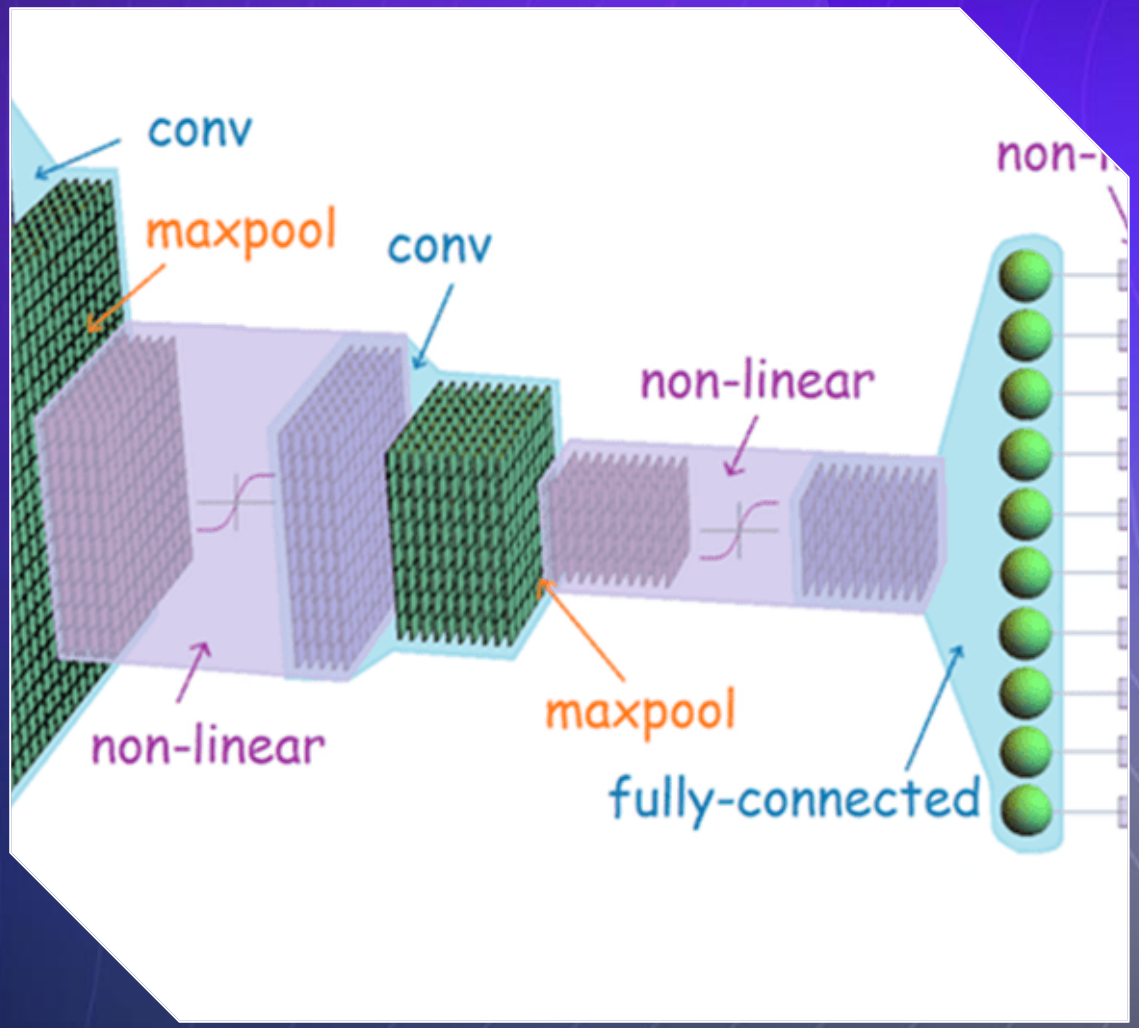
Separação entre Treino e Teste

- Dataset dividido em 90% para treino e 10% para teste.
- Uso de 'stratify' para manter classes equilibradas.
- One-hot encoding para transformar categorias em números.
- Verificação constante para evitar overfitting.



```python
# Divisão dos dados
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.1, stratify=labels, random_state=42)

# Codificar labels
num_classes = len(label_dict)
y_train_hot = to_categorical(y_train, num_classes=num_classes)
y_test_hot = to_categorical(y_test, num_classes=num_classes)

```

Estrutura da Rede Neural



DEFININDO A ARQUITETURA DA REDE

TREINAMENTO DA REDE

ANALISE DE RESULTADOS

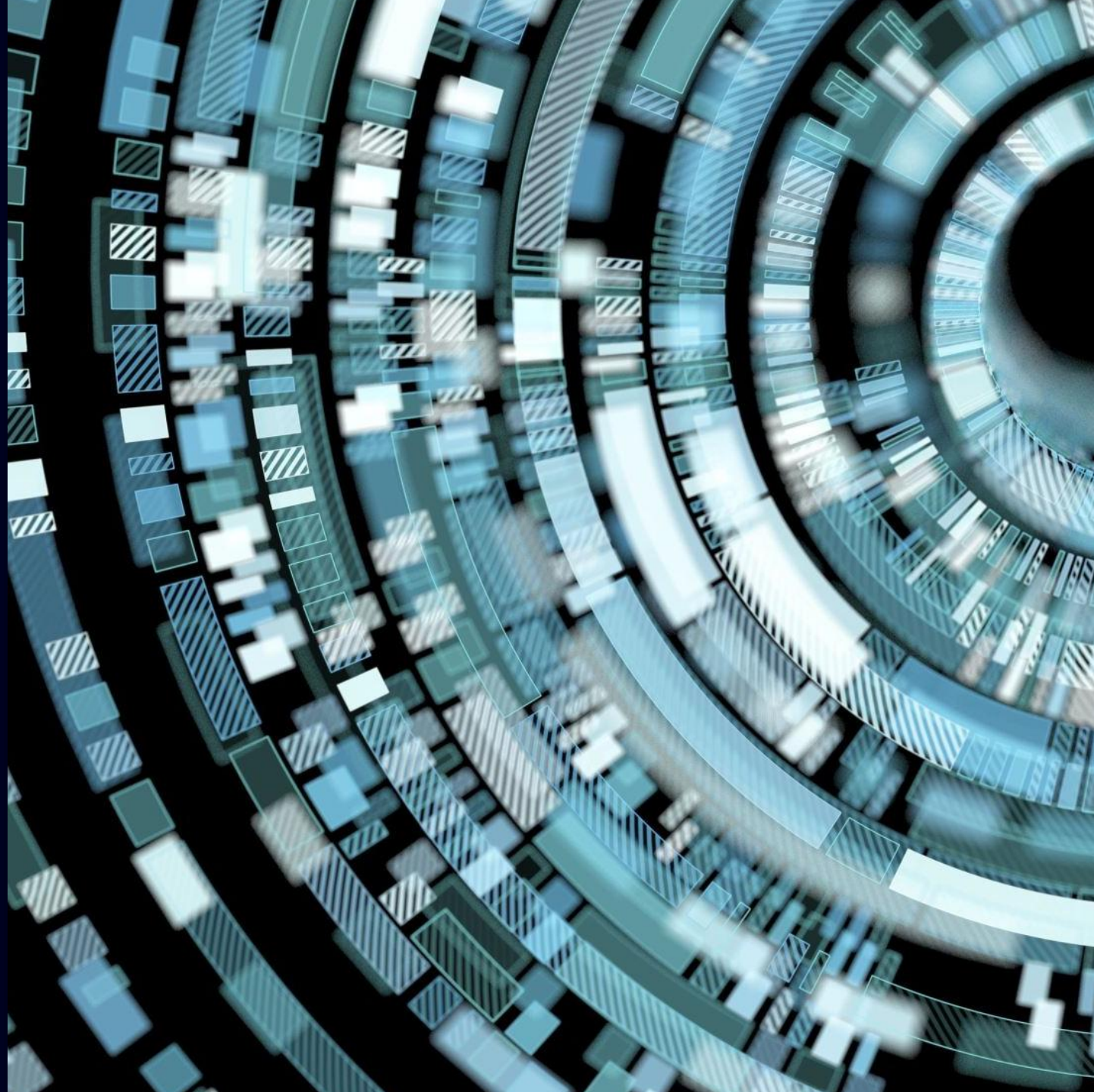
TESTE COM VIDOS CASEIROS

Definindo a arquitetura da rede

- **Entrada dos Dados**
  - Sequência de 60 frames, resolução 64x64, escala de cinza.

- **Blocos Convolucionais e Pooling**
  - **Bloco 1:** Conv3D (32 filtros) + ReLU + MaxPooling3D.
  - **Bloco 2:** Conv3D (64 filtros) + ReLU + MaxPooling3D.
  - **Bloco 3:** Conv3D (128 filtros) + ReLU + GlobalAveragePooling3D.

- **Camadas Densas**
  - **Densa 1:** 128 neurônios + Dropout (50%) para evitar overfitting.
  - **Saída:** Softmax para classificação dos sinais (10 classes).



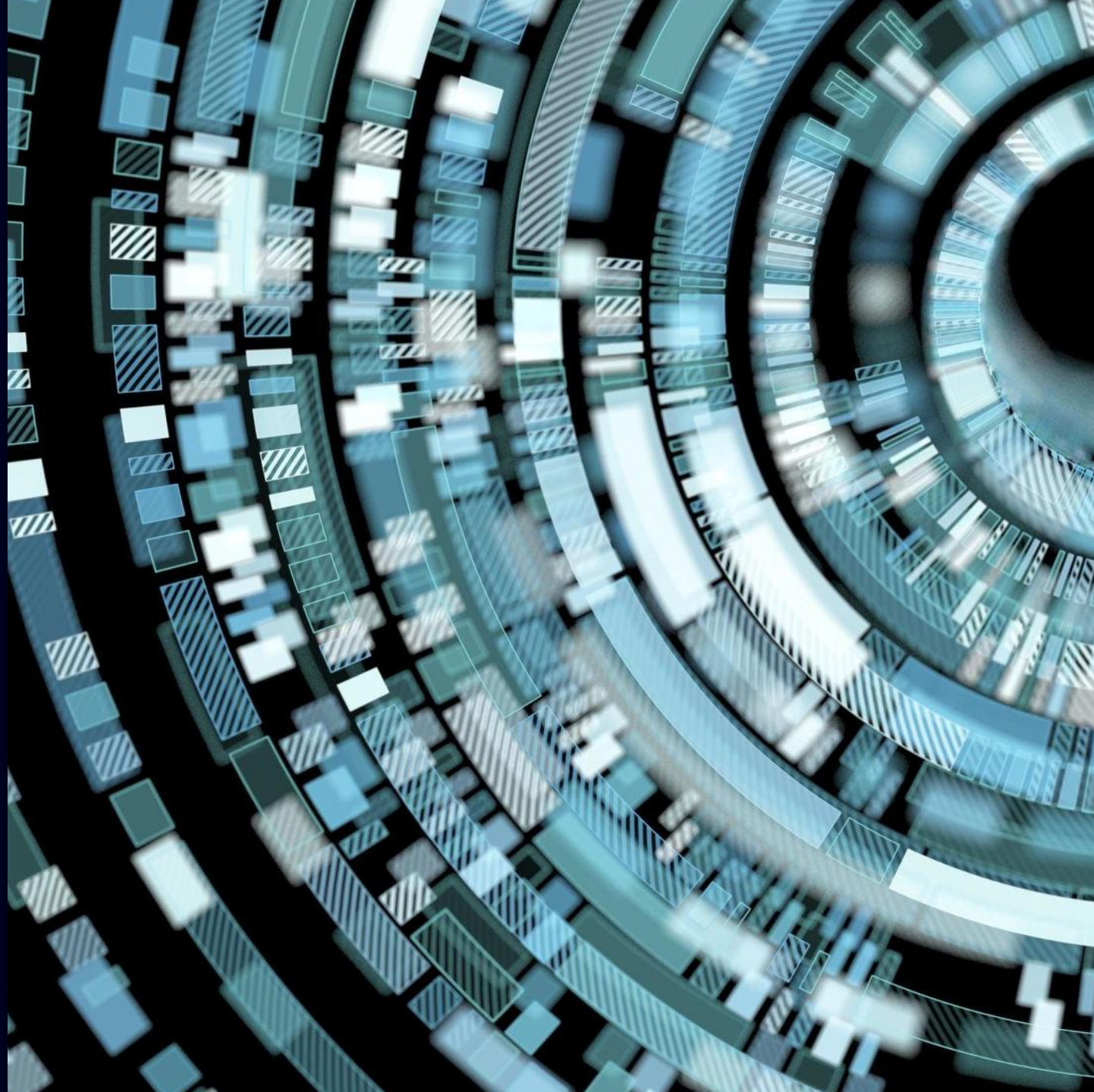
```python
def build_3d_cnn(input_shape=(num_frames, altura, largura, canais), num_classes=num_classes):
    model = models.Sequential([
        layers.Conv3D(32, (3, 3, 3), activation='relu', input_shape=input_shape),
        layers.MaxPooling3D((2, 2, 2)),
        layers.Conv3D(64, (3, 3, 3), activation='relu'),
        layers.MaxPooling3D((2, 2, 2)),
        layers.Conv3D(128, (3, 3, 3), activation='relu'),
        layers.GlobalAveragePooling3D(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model
```

Layer (type)	Output Shape	Param #
conv3d (Conv3D)	(None, 28, 62, 62, 32)	896
max_pooling3d (MaxPooling3D)	(None, 14, 31, 31, 32)	0
conv3d_1 (Conv3D)	(None, 12, 29, 29, 64)	55,360
max_pooling3d_1 (MaxPooling3D)	(None, 6, 14, 14, 64)	0
conv3d_2 (Conv3D)	(None, 4, 12, 12, 128)	221,312
global_average_pooling3d (GlobalAveragePooling3D)	(None, 128)	0
dense (Dense)	(None, 128)	16,512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

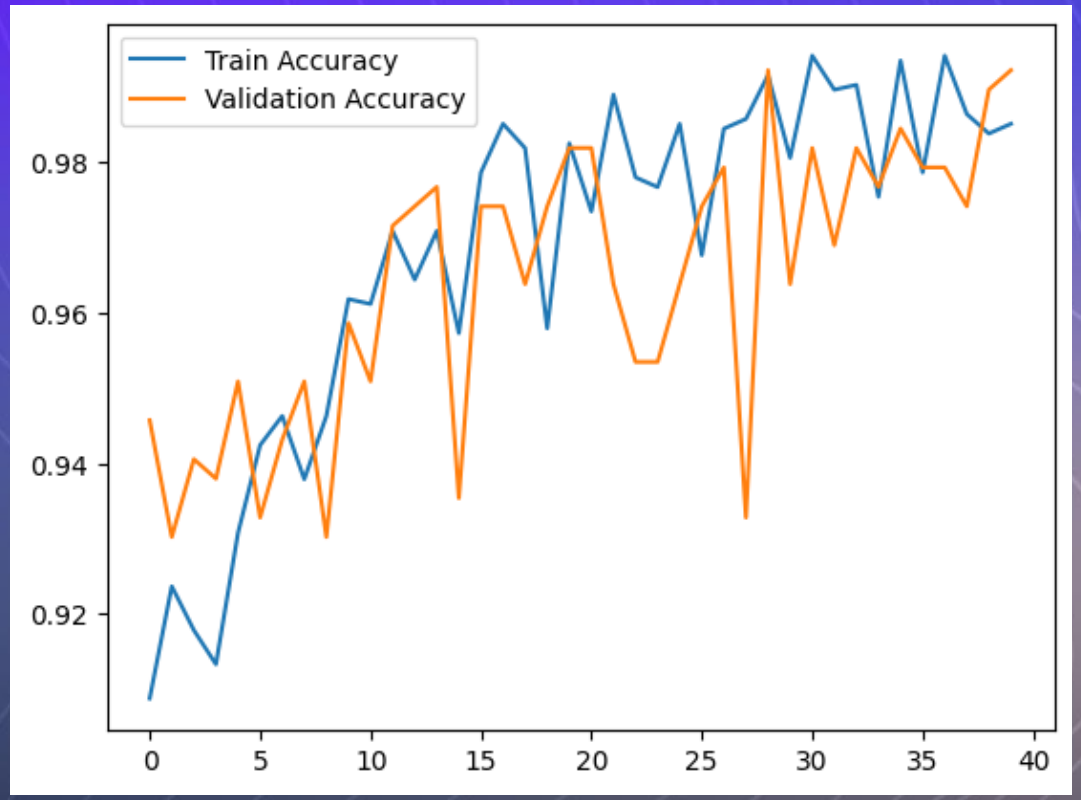
Total params: 295,370 (1.13 MB)

Treinamento da Rede Neural

- **Configurações do Treinamento**
  - Otimizador: Adam.
  - Função de perda: Categorical Crossentropy.
  - Métrica: Acurácia.
- **Processo de Treinamento**
  - Batch size: 8.
  - Número de épocas: 40.
- **Estratégias Adotadas**
  - Data augmentation para aumentar a variabilidade.
  - Early stopping para evitar overfitting.
- **Resultado**
  - Acurácia de 99% na validação.

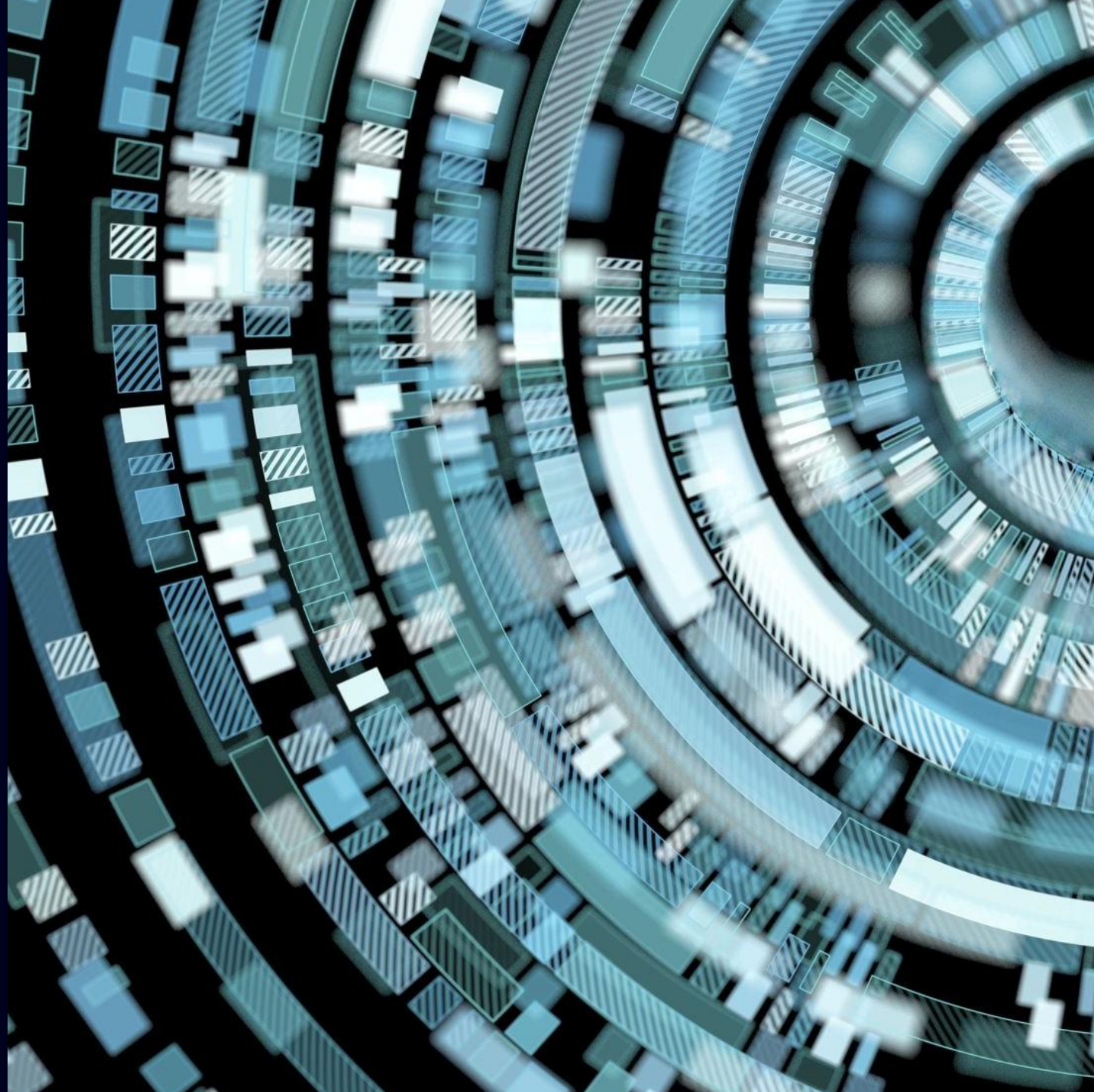


```python
# Treinamento
history = model.fit(X_train, y_train_hot, batch_size=8, epochs=40, validation_split=0.2)
```



Análise de resultados

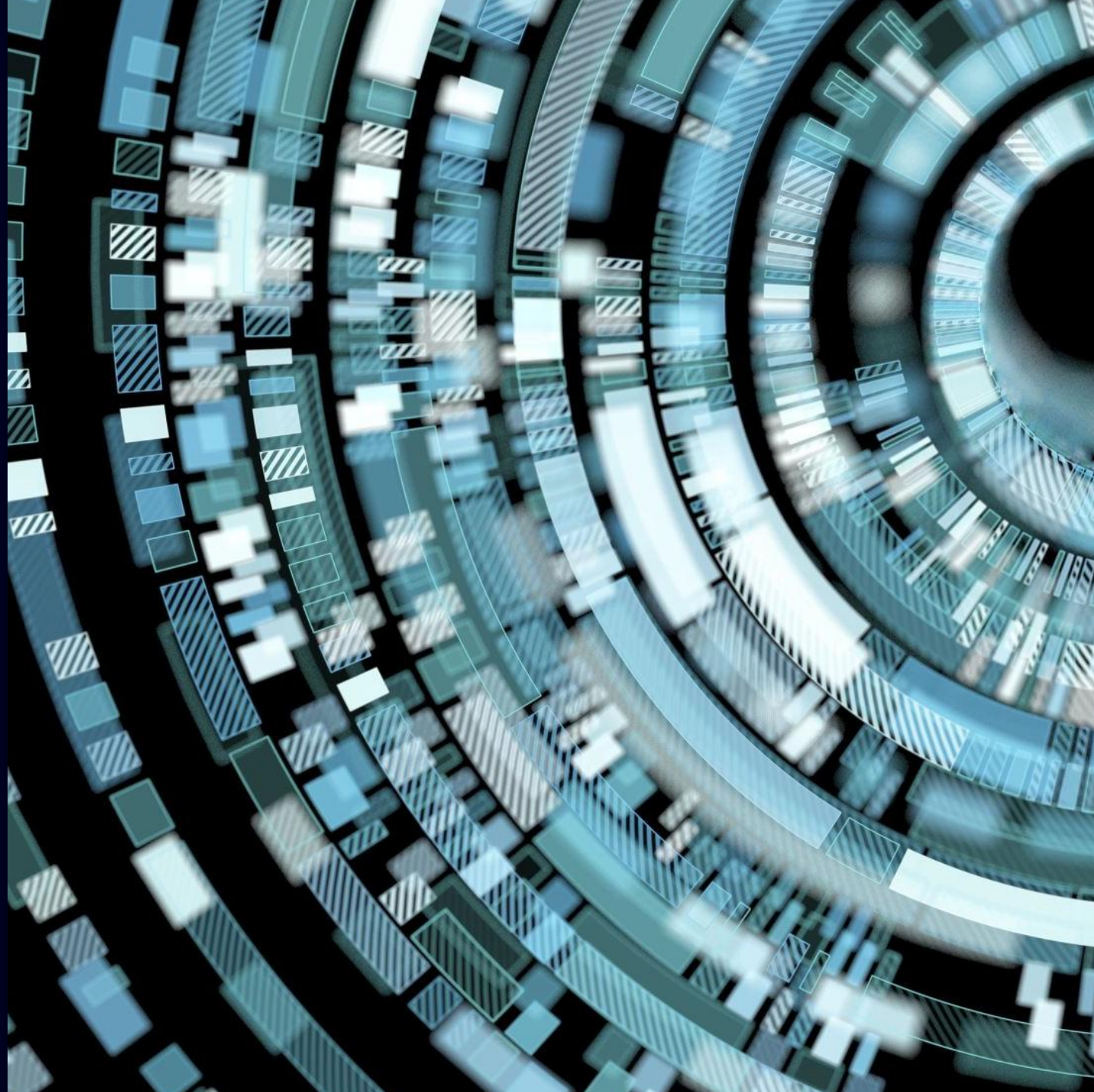
- **Acurácia Final**
  - Modelo atingiu 99% de acurácia na validação.
- **Desempenho nos Sinais**
  - Alta precisão na distinção dos 10 sinais de Libras.
- **Generalização**
  - Modelo apresentou bom desempenho com dados de teste não vistos.
- **Impacto da Padronização**
  - Consistência nos frames e resolução contribuiu para o desempenho.
- **Contribuição do Data Augmentation**
  - Melhorou a robustez do modelo contra variações nos vídeos.



```
Generating test predictions...
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 129ms/step
[[0. 0. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]
 ...
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]]
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 126ms/step - accuracy: 0.9925 - loss: 0.0182
Test loss: 0.022205634042620 66
Test accuracy: 0.9906976819038391
Predicted: 2 (expected: 2)
Predicted: 6 (expected: 6)
Predicted: 7 (expected: 7)
Predicted: 0 (expected: 0)
Predicted: 1 (expected: 1)
Predicted: 1 (expected: 1)
Predicted: 6 (expected: 6)
Predicted: 1 (expected: 1)
Predicted: 3 (expected: 3)
Predicted: 4 (expected: 4)
Predicted: 7 (expected: 7)
Predicted: 9 (expected: 9)
Predicted: 8 (expected: 8)
...
Predicted: 7 (expected: 7)
Predicted: 0 (expected: 0)
Predicted: 5 (expected: 5)
Predicted: 2 (expected: 2)
```

Teste com vídeos caseiros

- **Objetivo dos Testes**
  - Avaliar a eficácia do modelo com vídeos reais e não processados.
- **Configuração dos Vídeos**
  - Gravados em diferentes ambientes e condições de iluminação.
- **Desafios Enfrentados**
  - Movimentos imprecisos e variações no enquadramento.
- **Resultados Obtidos**
  - Modelo reconheceu sinais corretamente na maioria dos casos.
- **Observações Importantes**
  - Pequenas discrepâncias ocorreram devido à falta de padronização.
  - Reflete robustez, mas indica necessidade de ajustes para casos extremos.



```
[INFO] Classifying masked videos...
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
Video: mask_acontecer.mp4 - Predicted Class: 0 - Probabilities: [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Video: mask_amarelo.mp4 - Predicted Class: 1 - Probabilities: [[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Video: mask_america.mp4 - Predicted Class: 8 - Probabilities: [[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Video: mask_aproveitar.mp4 - Predicted Class: 3 - Probabilities: [[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Video: mask_banheiro.mp4 - Predicted Class: 4 - Probabilities: [[0.06 0.   0.   0.01 0.88 0.   0.   0.   0.   0.05]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Video: mask_barulho.mp4 - Predicted Class: 5 - Probabilities: [[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Video: mask_cinco.mp4 - Predicted Class: 3 - Probabilities: [[0.   0.   0.   0.6  0.   0.19 0.   0.21 0.   0.  ]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Video: mask_espelho.mp4 - Predicted Class: 7 - Probabilities: [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Video: mask_esquina.mp4 - Predicted Class: 8 - Probabilities: [[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Video: mask_medo.mp4 - Predicted Class: 9 - Probabilities: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

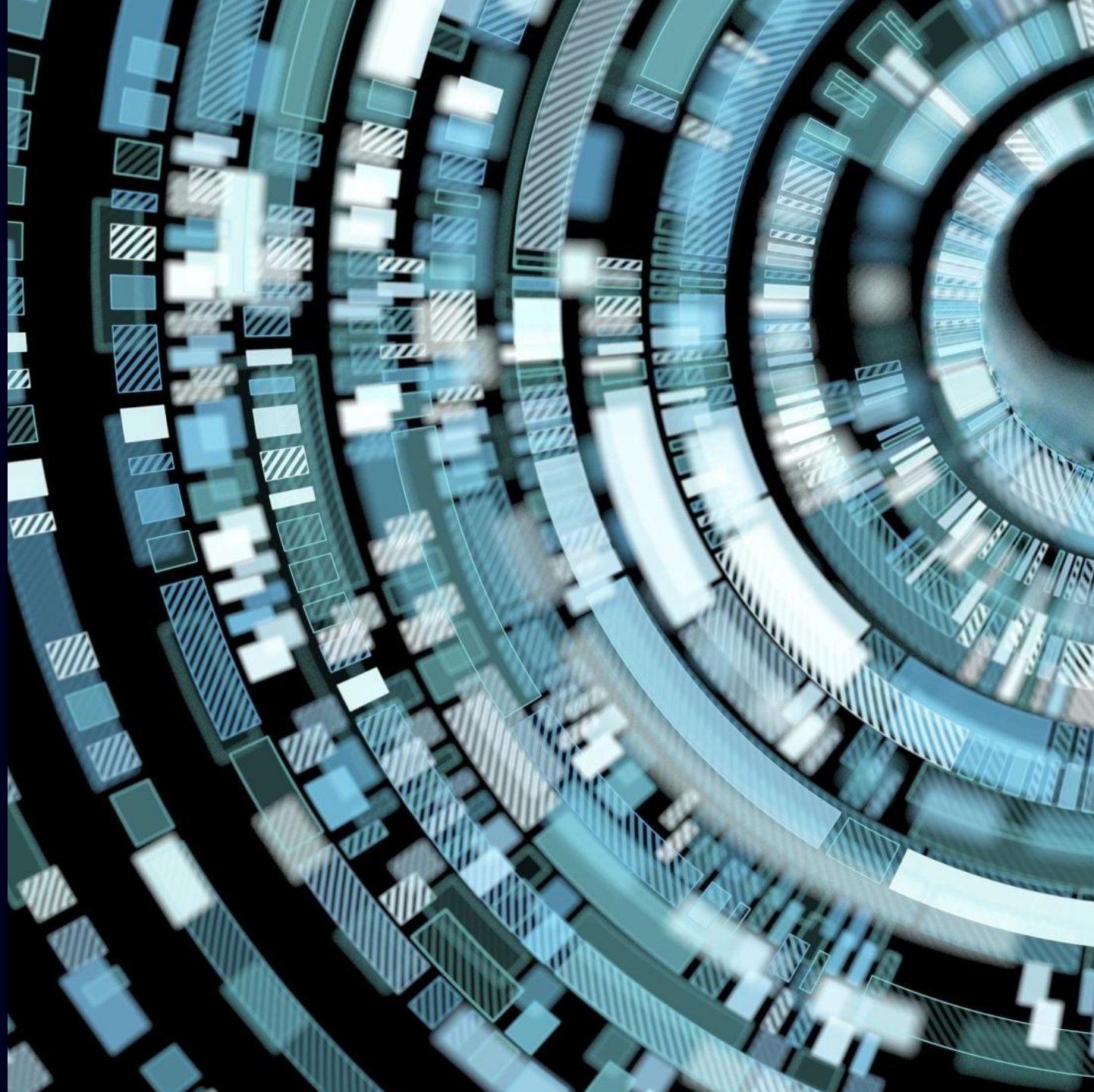
Reconhecimento em Tempo Real

```
nds = mp.solutions.hands
se = mp.solutions.pose
awing = mp.solutions.drawing_utils

regar o modelo treinado
_path = r"C:\Users\victor\OneDrive\Documentos\GitHub\ProjetoFinal_VisaoComp\model_mask1
 = load_model(model_path)

âmetros
a, largura, canais = 64, 64, 1  # Dimensões e canais
rames = 30  # Número de frames esperados
t_size = (largura, altura)
_rate = 30  # FPS da câmera

_labels = ['Acontecer', 'Amarelo', 'America', 'Aproveitar', 'Banheiro',
           'Barulho', 'Cinco', 'Espelho', 'Esquina', 'Medo']

ção para processar frames capturados
rocess_video_frames(frames, target_size, num_frames, model):
esized_frames = [cv.resize(frame, target_size) / 255.0 for frame in frames]

f len(resized_frames) > num_frames:
   indices = np.linspace(0, len(resized_frames) - 1, num_frames).astype(int)
   resized_frames = [resized_frames[i] for i in indices]
lif len(resized_frames) < num_frames:
   padding = num_frames - len(resized_frames)
   resized_frames.extend([resized_frames[-1]] * padding)

      ames = np.array(resized_frames).reshape((1, num_frames, *target_size, canais))
        = model.predict(input_frames)
        tion
```

LIBRAS DETECTION – PRESS TO READ

LIBRAS DETECTION – REAL TIME

DEMONSTRAÇÃO



Aplicações e Potencialidades

MAIOR INCLUSÃO

ACESSIBILIDADE DIGITAL

ATENDIMENTO AO CLIENTE

COMUNICAÇÃO EM AMBIENTES PÚBLICOS

ENTRETENIMENTO E MÍDIA

Referências

- SINGH, A.; SINGH, R. A review of machine learning algorithms for breast cancer detection. IEEE Access, v. 10, p. 31524-31535, 2022. DOI: 10.1109/ACCESS.2022.9739689. Disponível em: https://ieeexplore.ieee.org/document/9739689. Acesso em: 4 nov. 2024.

- TAVAKOLI, A. et al. The Language of Actions: Towards Robust Action Recognition from Videos. arXiv preprint arXiv:2204.03328, 2022. Disponível em: https://ar5iv.labs.arxiv.org/html/2204.03328. Acesso em: 4 nov. 2024.

- LIM, S. et al. Human Action Recognition Based on 3D Temporal Convolutional Networks and Transfer Learning. IEEE Access, v. 11, p. 84548-84559, 2023. DOI: 10.1109/ACCESS.2023.10004413. Disponível em: https://ieeexplore.ieee.org/document/10004413. Acesso em: 4 nov. 2024.

- FUSETI, Alexandre. Desenvolvimento de um sistema de reconhecimento de gestos em Libras utilizando visão computacional. 2024. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2024. Disponível em: https://sites.uel.br/dc/wp-content/uploads/2024/08/PROJETO_TCC_Alexandre_Fuzeti.pdf. Acesso em: 4 nov. 2024.

- SOUSA, Fábio Gomes de. Reconhecimento de Gestos em Libras Usando Redes Neurais Convolucionais e Dados de Profundidade. 2023. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, 2023. Disponível em: https://www.ppgee.ufmg.br/defesas/1393D.PDF. Acesso em: 4 nov. 2024.