PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO - PROPESP
COORDENAÇÃO DE PESQUISA E PROJETOS INSTITUCIONAIS - CPPI
PROGRAMA DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA DA UNIVERSIDADE DO ESTADO DO AMAZONAS

RELATÓRIO MENSAL			
EDITAL ICT	(X) Edital Nº 031/2025 - GR/UEA		
TIPO DE PROJETO	( ) PBICT-Af/UEA (X) PAIC/FAPEAM ( ) PIBIC/CNPq ( ) PIBITI/CNPq ( ) PIBIC Af/CNPq ( ) VOLUNTÁRIO	**SISPROJ:**	59617
TÍTULO DO PROJETO:	Sistema de Detecção de Quedas Utilizando Inteligência Artificial e Alerta Automatizado		
ALUNO:	Nelson Emeliano Silva		
ORIENTADOR:	Angilberto Muniz Ferreira Sobrinho		
MÊS DE REFERÊNCIA DO RELATÓRIO	Outubro/2025	**DATA DE ENTREGA DO RELATÓRIO**	28/10/2025

ATIVIDADES DESENVOLVIDAS

1. Resumo das Atividades do Período

Durante esse mês, o foco foi efetivamente na fase de desenvolvimento e experimentação. O esforço foi concentrado na codificação da arquitetura do modelo de IA e na validação de todo o pipeline de dados. Foram executados os primeiros ciclos de treinamento para estabelecer um baseline de performance, confirmando que os dados processados alimentam a rede neural sem erros de formato.

2. Detalhamento das Atividades Realizadas

2.1. Implementação da Arquitetura do Modelo (CNN-LSTM)

A primeira versão da arquitetura híbrida foi codificada em Python, utilizando a biblioteca TensorFlow/Keras. O modelo utiliza a técnica de *transfer learning*, aproveitando a **MobileNetV2** (pré-treinada no ImageNet) como um extrator de características espaciais. Os pesos desta base foram "congelados" (trainable = False).

A camada TimeDistributed foi usada para aplicar a CNN em cada quadro de uma sequência de vídeo. Na sequência, uma camada LSTM analisa os padrões temporais dessas características para, enfim, uma camada Dense com ativação sigmoid realizar a classificação binária (Queda / Não Queda).

O *snippet* de código abaixo ilustra a definição da arquitetura:

```python
class FallDetectionModel:
    def build_model(self):
        # Input layer para sequências de vídeo
        input_layer = layers.Input(
            shape=(self.config['sequence_length'],) + self.config['input_shape'],
            name='video_input'
        )

        # MobileNetV2 como base CNN (pré-treinada no ImageNet)
        base_model = MobileNetV2(
            input_shape=self.config['input_shape'],
            include_top=False,
            weights='imagenet'
        )

        # Congelar pesos da MobileNetV2
        base_model.trainable = False

        # Aplicar CNN em cada frame da sequência
        cnn_features = layers.TimeDistributed(
            base_model,
            name='cnn_extractor'
        )(input_layer)

        # Global Average Pooling para cada frame
        pooled_features = layers.TimeDistributed(
            layers.GlobalAveragePooling2D(),
            name='global_pooling'
        )(cnn_features)

        # Dropout para regularização
        dropout_features = layers.TimeDistributed(
            layers.Dropout(self.config['dropout_rate']),
            name='cnn_dropout'
        )(pooled_features)
```

2.2. Validação do Pipeline de Pré-processamento

O script de pré-processamento foi finalizado para carregar os vídeos dos datasets coletados e transformá-los no formato de entrada exigido pelo modelo ((N_sequencias, 20, 224, 224, 3)).

A lógica central do script utiliza a biblioteca OpenCV (cv2) para ler os arquivos de vídeo, extrair os quadros (frames), aplicar o redimensionamento e empilhá-los em sequências.

O snippet abaixo demonstra a lógica de processamento do video:

```python
def process_video_file(self, output_path=None, display=True):
    """
    Processa arquivo de vídeo completo
    """
    if not self.cap or not self.cap.isOpened():
        self.logger.error("Vídeo não carregado")
        return

    # Configurar writer de vídeo se necessário
    writer = None
    if output_path:
        fourcc = cv2.VideoWriter_fourcc(*'XVID')
        fps = self.cap.get(cv2.CAP_PROP_FPS)
        width = int(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        writer = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

    frame_count = 0
    total_frames = int(self.cap.get(cv2.CAP_PROP_FRAME_COUNT))

    self.logger.info(f"Iniciando processamento de {total_frames} frames...")
```

2.3. Execução do Treinamento Inicial (Baseline)

Com o modelo construído e os dados sendo processados corretamente, foram executados os primeiros ciclos de treinamento (model.fit()). O objetivo não foi obter a acurácia máxima, mas sim estabelecer um baseline e confirmar a funcionalidade do pipeline.

O modelo foi compilado com o otimizador Adam e a função de perda binary_crossentropy, e o treinamento foi iniciado com uma subamostra dos dados.

```python
def save_model(self, filepath=None):
    """
    Salva o modelo treinado

    Args:
        filepath: Caminho para salvar (opcional)
    """
    if self.model is None:
        raise ValueError("Modelo não foi treinado")

    if filepath is None:
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        filepath = os.path.join(self.models_dir, f'fall_detection_model_{timestamp}.h5')

    self.model.save_model(filepath)
    self.logger.info(f"Modelo salvo em: {filepath}")

    return filepath
```

O treinamento inicial demonstrou convergência, com a métrica de perda (loss) diminuindo nas primeiras épocas, validando com sucesso a arquitetura e o fluxo de dados.

3. Resultados do Período

- **Modelo de IA (Versão 0.1):** Foi produzido um script Python funcional com a arquitetura CNN-LSTM compilada.
- **Pipeline de Dados Validado:** Foi confirmado que os scripts de pré-processamento convertem os vídeos brutos em lotes de dados no formato correto para o TensorFlow.
- **Métricas de Baseline Estabelecidas:** Foram obtidas as primeiras métricas de acurácia e perda em um subconjunto de dados, que servirão como ponto de partida para as futuras otimizações do modelo.

AVALIAÇÃO DO ORIENTADOR

Deverá ser preenchido um relatório mensal pelo (a) aluno (a) e orientador (a), informando as atividades desenvolvidas e anexado no SISPROJ até o dia 30 de cada mês, a partir do mês de implementação do projeto. O acompanhamento do relatório e frequência mensais será realizado pelo Comitê Local.