Detecção da presença de capacetes de segurança

Gean Lucas Ramos Vieira	2018004881
Paulo Henrique Faria Arruda	2018007874
Pedro Henrique Rodrigues	2018015830
Thiago Crabi de Freitas	2018014565

Universidade Federal de Itajubá

1 Objetivos

Objetiva-se nesse projeto criar um equipamento capaz de realizar a detecção de capacetes de segurança em indivíduos. Para tanto, deve-se captar uma imagem do mesmo como entrada para o modelo de inferência, para que o mesmo possa realizar a identificação do EPI, podendo trabalhar em conjunto com outros equipamentos para a emissão de alertas para o setor de segurança do trabalho.

2 Descrição

De acordo com o G1, o Brasil ocupa a segunda posição no ranking de mortalidade por acidentes de trabalho, em comparação com os demais países que compõem o G20. Segundo Luís Fabiano de Assis, procurador do Ministério Público do Trabalho, acidentes e doenças laborais foram responsáveis pela perda de R$ 300 bilhões do PIB brasileiro em 2020. Ademais, existe ainda o impacto ao indivíduo, tanto no aspecto físico quanto psicológico, que pode impactar no desenvolvimento de suas atividades cotidianas. Sendo assim, o desenvolvimento de novas tecnologias e práticas nesse contexto devem ser incentivadas, de modo a reduzir os danos que decorrem dele.

Nesse sentido, apresenta-se nesse trabalho o desenvolvimento de um modelo para a identificação da presença de capacetes de segurança em indivíduos. Com isso, espera-se incentivar a utilização desses equipamentos de proteção individuais, através da possibilidade de obtenção de um equipamento de fiscalização de baixo custo.

Para isso, tem-se a captação de um grande conjunto de imagens dividas em duas categorias: presença e ausência, de acordo com a identificação ou não de um capacete na cabeça do indivíduo. Em posse disso, treina-se um modelo de redes neurais e obtém-se um

modelo de inferência, que poderá ser utilizado em ambientes hostis e também em locais de difícil acesso, tais como no setor de construção civil e de mineração.

Espera-se que o modelo treinado seja implementado em equipamentos como os kits recebidos na disciplina, compostos por sensores, microcontroladores e módulos de conexão. Dessa forma, uma vez que um indivíduo sem capacete seja detectado, pode-se emitir um alerta ao responsável do setor, que poderá alertar o trabalhador e, em seguida, tomar as medidas cabíveis.

3 Hardware

Para este projeto será utilizado o arduino 33 BLE sense, que possui um 32-bit ARM® Cortex™-M4 CPU rodando a 64 MHz, tem 1MB de memória flash e 256KB de SRAM. Será utilizado ainda a câmera que o arduino possui, que é uma câmera OV7675 640x480, para a implementação final do projeto.

4 Coleta de Dados

Para o desenvolvimento deste projeto foram utilizados dois datasets: o Hard Hat Workers e o Scut-Head. O dataset Hard Hat Workers é composto por 5.000 imagens capturadas em diferentes ambientes de trabalho na China, sobretudo em construções e fábricas onde o capacete de segurança geralmente é um item de uso obrigatório. O dataset inclui imagens das classes Helmet (capacete) e Head (cabeça), sendo utilizado originalmente para fazer detecção e não para classificação. A Figura 1 mostra exemplos de imagens contidas no dataset.

Figura 1: Exemplos de imagens contidas no dataset Hard Hat Workers.

O Scut-Head é um dataset desenvolvido para a detecção de cabeças, sendo formado por 4.405 imagens marcadas com 111.251 cabeças. O dataset é dividido em duas partes,

a PartA e a PartB. A PartA inclui 2.000 imagens obtidas a partir de vídeos de câmeras de monitoramento de salas de aula de uma universidade, com 67.321 cabeças marcadas. A PartB é formada por 2.405 imagens rastreadas da internet, possuindo 49.930 cabeças marcadas. Neste projeto foi utilizado a PartB do dataset Scut-Head. A Figura 2 mostra exemplos de imagens contidas no dataset.

Figura 2: Exemplos de imagens contidas no dataset Scut-Head.

5 Pré-processamento

Na fase de pré-processamento, os dados precisam ser trabalhados no intuito de obter modelos melhores. No caso, dois testes foram feitos: dataset com imagens de tamanho igual a 48×48 pixels e outro com o tamanho das imagens igual a 96×96 pixels. Cada imagem está associada a um arquivo XML, o qual contém a posição em que se encontra as cabeças e o respectivo label (helmet ou head). Nesse sentido, o arquivo XML foi utilizado para realizar os recortes nas imagens.

No primeiro caso, isto é, com as imagens de tamanho igual a 48×48 pixels, as seguintes ações foram tomadas:

- As imagens foram cortadas exatamente nas áreas onde estavam as cabeças;
- O tamanho das imagens foram normalizados para 48×48 pixels;
- As imagens que continham cabeças muito pequenas foram descartadas;
- Após as etapas anteriores, é visto que os dados obtidos a partir do dataset Hard Hat Workers possuíam mais helmet do que head. Nesse sentido, o dataset Scut-Head foi utilizado para complemento. Com isso, para cada classe há 5.416 imagens.

Para o segundo caso, isto é, com as imagens de tamanho igual a 96×96 pixels, as seguintes ações foram tomadas:

- Utilização de data augmentation. No caso, as seguintes ações foram realizadas: variação do zoom, mudança da posição da área de recorte e flip horizontal aleatório, conforme Figura 4;

Figura 3: Processo para a obtenção do dataset de imagens 48×48.

- O tamanho das imagens foram normalizados para 96×96 pixels;
- As imagens que continham cabeças muito pequenas foram descartadas;
- Novamente o dataset Scut-Head foi utilizado para complementar os dados obtidos a partir do dataset Hard Hat Workers. Com isso, para cada classe há 4.773 imagens.

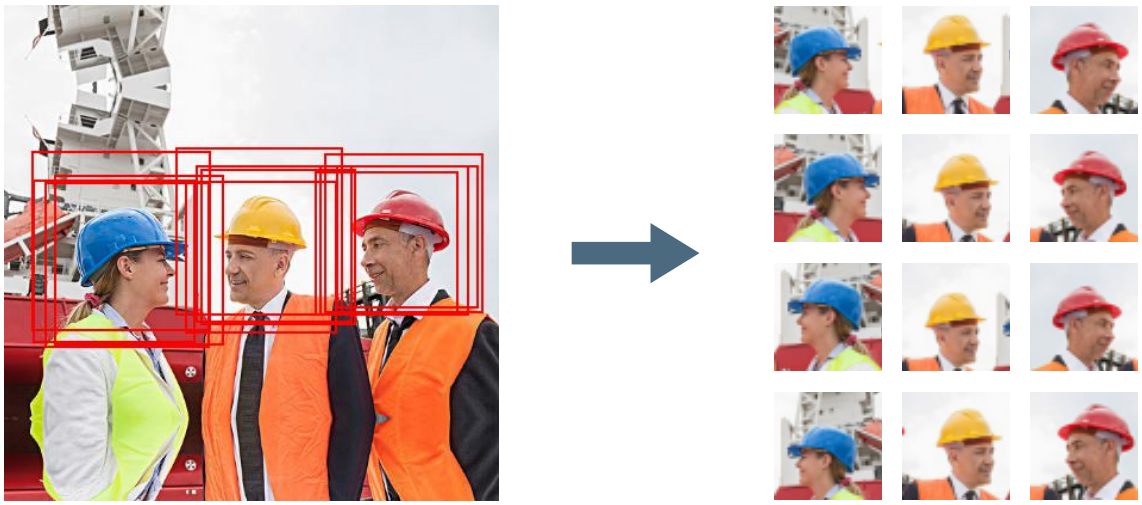

Figura 4: A figura à esquerda mostra os recortes feitos na imagem e a figura à direita mostra os resultados obtidos. Nesse processo foi utilizado a técnica de data augmentation.

Os códigos utilizados para realizar o pré-processamento são apresentados abaixo.

```
def img_augmentation(img, label, bnd_box, final_size):
    data = []
    width = bnd_box[2] - bnd_box[0]

    for _ in range(4):
        size = randint(int(width*1.3), int(width*1.6))
```

```
            x =  randint(max(0, bnd_box[2]-size), bnd_box[0])
            y =  randint(max(0, bnd_box[3]-size), bnd_box[1])
            result = img.crop([x, y, x + size, y + size])
            result = result.resize((final_size, final_size), Image.ANTIALIAS)
            if random() > 0.5:
                result = result.transpose(Image.FLIP_LEFT_RIGHT)
            if (x + size) < img.width and (y + size) < img.height:
                data.append((
                    label,
                    result
                ))

    return data

def crop_objects(filename, size):
    with open(f'archive/annotations/{filename}.xml') as f:
        data = BeautifulSoup(f, 'lxml')
    img = Image.open(f'archive/images/{filename}.png')
    imgs = []
    for obj in data.find_all('object'):
        bnd_box = [int(i.get_text()) for i in obj.find('bndbox')
            if isinstance(i, bs4.element.Tag)]
        width, height = bnd_box[2] - bnd_box[0], bnd_box[3] - bnd_box[1]
        proportion = width/height
        if proportion > 0.8 and proportion < 1.25:
            if width > size*0.75:
                if width > size*0.9:
                    imgs.append((
                        obj.find('name').get_text(),
                        img.crop(bnd_box).resize((size, size), Image.ANTIALIAS)
                    ))
                imgs.extend(img_augmentation(
                    img, obj.find('name').get_text(),
                    size, bnd_box
                ))

    return imgs
```

6 Model Design

Foram desenvolvidos três modelos diferentes onde variou-se a arquitetura, o dataset utilizado e o formato de entrada. O modelo 1 utiliza o dataset com imagens de tamanho 48×48 pixels e em Grayscale, enquanto que o modelo 2 tem como entrada imagens com tamanho de 32×32 pixels e em RGB. As arquiteturas dos modelos 1 e 2 podem ser vistas na Figura 5. O modelo 3 utiliza como entrada o dataset com imagens de tamanho 96×96 pixels e em RGB. Nesse modelo, foi aplicado Transfer Learning no dataset, sendo utilizado o modelo MobileNetV1 0.25, sem um layer denso no final e com Dropout de 0.1.

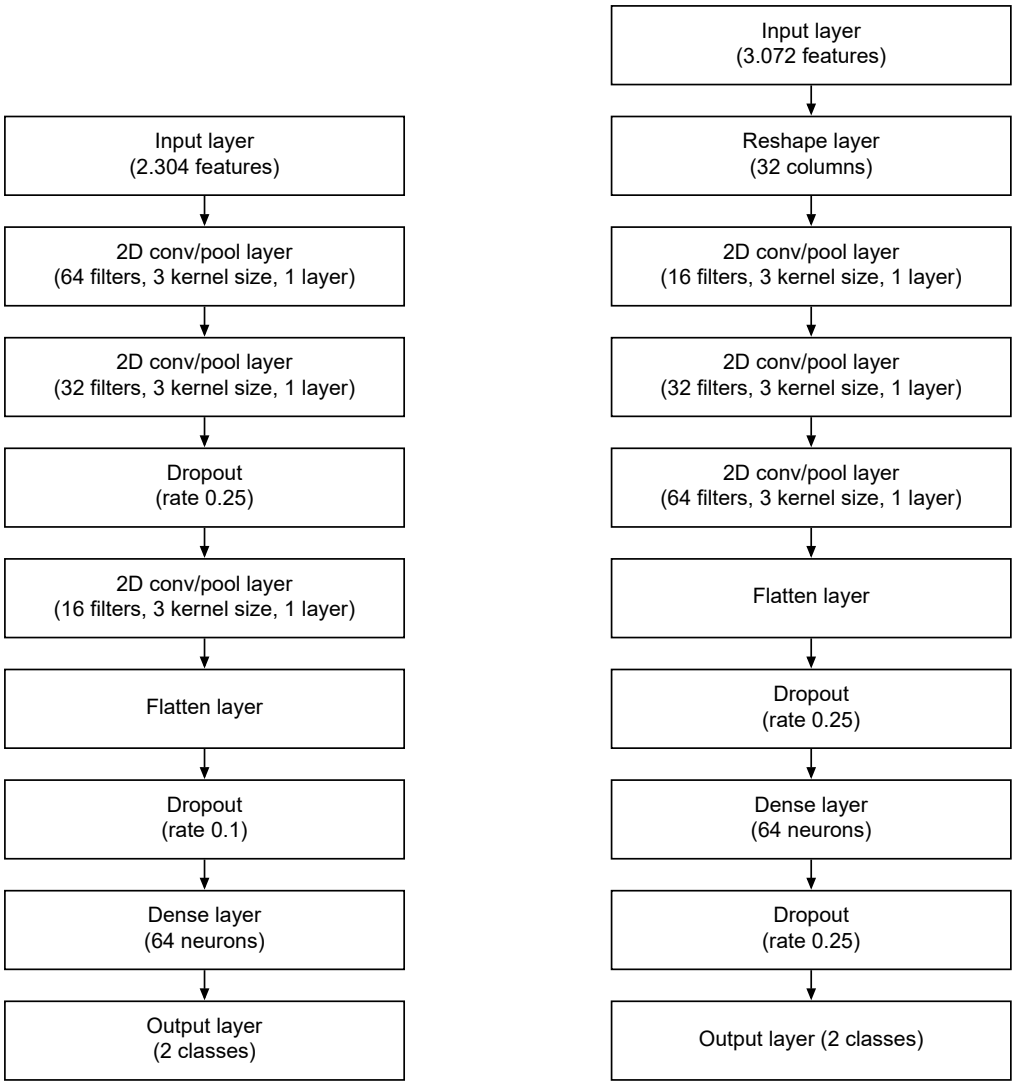

Figura 5: Arquitetura dos modelos utilizados. (a) Modelo 1. (b) Modelo 2.

7 Resultados

Feito o treinamento dos três modelos, as matrizes de confusão de teste foram obtidas e são mostradas nas Tabelas 1, 2 e 3. À primeira vista, percebe-se que o modelo 1 não apresenta uma acurácia alta se comparado aos modelos 2 e 3. Por outro lado, o modelo 2 e 3 são bem próximos.

A fim de realizar uma comparação, a Tabela 4 foi gerada. É visto que, de fato, o modelo 1 apresentou uma acurácia mais baixa; entretanto, ele se sobressai em termos de velocidade. Os modelos 2 e 3 são próximos, conforme citado anteriormente, com o

modelo 2 possuindo uma vantagem em relação ao pico de RAM e o modelo 3 com relação à latência.

	Head	**Helmet**	**Uncertain**
Head	91.3%	6.0%	2.8%
Helmet	4.8%	91.1%	4.1%
F1 Score	0.93%	0.92%	

Tabela 1: Matriz de confusão – Modelo 1.

	Head	**Helmet**	**Uncertain**
Head	96.7%	2.7%	0.6%
Helmet	2.6%	96.7%	0.7%
F1 Score	0.97%	0.97%	

Tabela 2: Matriz de confusão – Modelo 2.

	Head	**Helmet**	**Uncertain**
Head	97.4%	1.4%	1.1%
Helmet	2.3%	96.4%	1.3%
F1 Score	0.98%	0.97%	

Tabela 3: Matriz de confusão – Modelo 3.

Modelo	**ACC Validação**	**ACC Teste**	**Latência**	**Pico de RAM**
1	93.0%	91.17%	1.807 ms	187.6 KB
2	97.7%	96.72%	513 ms	26.0 KB
3	97.4%	96.71%	321 ms	130.7 KB

Tabela 4: Comparação entre os três modelos.

A escolha do modelo dependerá da aplicação: se a velocidade é um fator crítico, mas a acurácia nem tanto, o modelo 1 pode ser utilizado. Por outro lado, se a acurácia é um fator crítico, mas a velocidade nem tanto, o modelo 2 e 3 podem ser utilizados.

8 Deploy

Foi feito o deploy utilizando a biblioteca arduino gerada pelo Edge Impulse, e a biblioteca TinyMLShield. A classificação é mostrada na porta serial e também indicada pelos LEDs: se for classificado como head, o LED ficará vermelho; se for classificado como helmet, o LED ficará verde. Além disso, caso os dois valores sejam próximos, os LEDs ficam apagados.

```
setup () {
    pinMode(LEDR, OUTPUT);
    pinMode(LEDG, OUTPUT);
    // ...
}

loop () {
    //...
    // acende vermelho se maior 0.7
    digitalWrite(LEDR, result.classification[0].value < 0.7);
    // acende verde se maior 0.7
    digitalWrite(LEDG, result.classification[1].value < 0.7);
    //...
}
```

9 Principais Desafios

Com relação ao equipamento, o uso do ESP-CAM foi considerado; entretanto, o mesmo apresentou erro no momento da inferência, conforme indicado abaixo.

```
ERR: failed to allocate tensor arena
Failed to allocate TFLite arena (error code 1)
Failed to run impulse (-6)
Predictions (DSP: 0 ms., Classification: 0 ms., Anomaly: 0 ms.):
```

Com relação aos datasets, enfrentou-se algumas dificuldades com um número maior de imagens de pessoas com capacete (classe Helmet) do que cabeças (Head), o que levou ao uso de outro dataset, o Scut-Head, para complementar as imagens de cabeças. Outro

problema enfrentado foi a baixa resolução apresentada em muitas imagens de cabeça, o que dificultou o uso no dataset de imagens de 96×96 pixels. Por fim, o fato de boa parte do dataset ser de pessoas asiáticas pode ter enviesado o modelo.

A última dificuldade enfrentada foi com relação ao Arduino. Houve dificuldade em posicionar corretamente a câmera por conta do corte que é feito para ajustar o tamanho da imagem.

10 Oportunidades de Melhoria

Embora os resultados dos modelos apresentados tenham se mostrado positivos, entende-se que existe uma margem de melhoria para o projeto. Nesse sentido, a fim de aumentar a robustez e a eficiência dos mesmos, propõe-se os seguintes pontos:

- Geração de dataset com GNU e montagem;
- Nova Classe;
- Ajuste de Hiperparâmetros.

Quanto ao primeiro ponto, deseja-se gerar um novo conjunto de imagens, através da sobreposição de capacetes de segurança em imagens que contenham indivíduos que não os estejam utilizando. Dessa forma, espera-se que o modelo possa ser melhor treinado e, portanto, capaz de realizar inferências com melhores resultados.

Já quanto ao segundo ponto, identificou-se a necessidade de criação de uma nova classe para o modelo, de modo que a condição em que não existe nenhuma pessoa na frente da câmera possa ser identificada. Com isso, é possível expandir a utilização do projeto, já que o mesmo pode, a partir disso, operar em períodos cíclicos de tempo, eliminando a necessidade de acionamento para a realização da inferência.

Por fim, com respeito ao terceiro ponto, tem-se a possibilidade de realizar mudanças no modelo, a fim de mitigar eventuais vieses e aumentar a sua taxa de acerto e também reduzir a sua latência. Dessa forma, torna-se possível fazer um melhor uso dos dados de entradas, garantindo que seja realizada a extração do maior número possível de informações de cada um deles.

11 Conclusão

Em posse das informações apresentadas, é possível concluir que os modelos escolhidos se mostraram promissores, tendo em vista os resultados apresentados. Ademais, como a situação do país é crítica com relação à situações dos acidentes de trabalho, a execução desse projeto possui um caráter social considerável, dado o potencial de aumento da qualidade das operações de fiscalização em ambientes de trabalho hostis, o que pode por sua vez contribuir para uma diminuição na quantidade de acidentes decorrentes da não utilização de capacetes.

Sabendo disso, espera-se implementar em um futuro próximo as melhorias supracitadas, além de realizar testes com outros dispositivos, tais como ESP, de modo a aumentar a qualidade do projeto e, com isso, fornecer uma solução viável para a sociedade, auxiliando na transformação dos ambientes de trabalho.

Referências

[1] https://makeml.app/datasets/hard-hat-workers

[2] https://sites.google.com/g.harvard.edu/tinyml/finalprojects

[3] https://github.com/HCIILAB/SCUT-HEAD-Dataset-Release