IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Detection Of ISI Mark On Helmet UsingSupervised ML

Anil Behal
Department of CSE, Chandigarh university

ABHISHEK KUMAR
Department of CSE, Chandigarh university

HEMANT YADAV
Department of CSE, Chandigarh university

ASHISH MISHRA
Department of CSE, Chandigarh university

HARMAN SOKHAL
Department of CSE, Chandigarh university

***Index Terms*—CNN , Tensorflow , Keras**

I. ABSTRACT

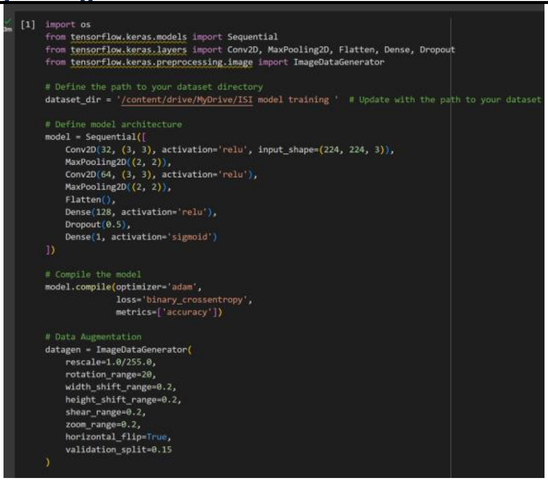
Road safety is becoming a bigger problem, and the repeated disastrous effects of collisions and injuries emphasise how important it is to take action to lower these risks. Acceptance of the helmet's vital role in preventing head injuries significant part in this safety framework. obligatory legal requirements to protect the quality and helmet safety emphasise the significance of ISI labelling as a sign on helmet. In this research work, we use supervised machine learning to detect the ISI mark on a helmet. adherence to these guidelines.

II. INTRODUCTION

The increasing concern about road safety, especially the repetitive catastrophic consequences of accidents and injuries, highlights the critical need for effective measures to reduce such risks Helmet acceptance that its critical importance in preventing head plays an important role in this safety model. Mandatory regulatory standards to maintain the quality and safety of the helmet highlight the importance of ISI marking as an indication of compliance with these standards but manual inspection with ISI markings poses challenges including the possibility of human error and inefficiency. Recognizing the importance of an improved and automated system, this study requires the development of a model for automatically detecting ISI symbols in helmet images By this special feature of helmets by addressing the quality control, the proposed system aims to increase accuracy, speed and reliability.

III. Literature Review

The motivation behind the automatic ISI mark recognition system stems from the pressing need to make existing methods of helmet quality assurance helmets act as a primary means of protection against accidental head injuries, making their quality, safety standards paramount Certification of compliance with regulatory standards It is important to ensure the effectiveness of the helmet in saving lives in. However, current manual evaluation [**?**] methods are prone to human error, subjectivity, and time inefficiencies. The illustrated system attempts to overcome these limitations by leveraging the power of computer vision and machine learning [**?**] [**?**]. Automation not only reduces the margin of error but also greatly speeds up the inspection process, resulting in increased throughput without compromising accuracy [**?**]This solves the challenges of manual inspection and contributes to improvements all come within the confidence of verifying ISI signals. [**?**] Thus, the motivation for this research stems from the search for technical solutions to improve the efficiency and effectiveness of helmet quality assessment procedures, ultimately improving road safety. [**?**] The analysis



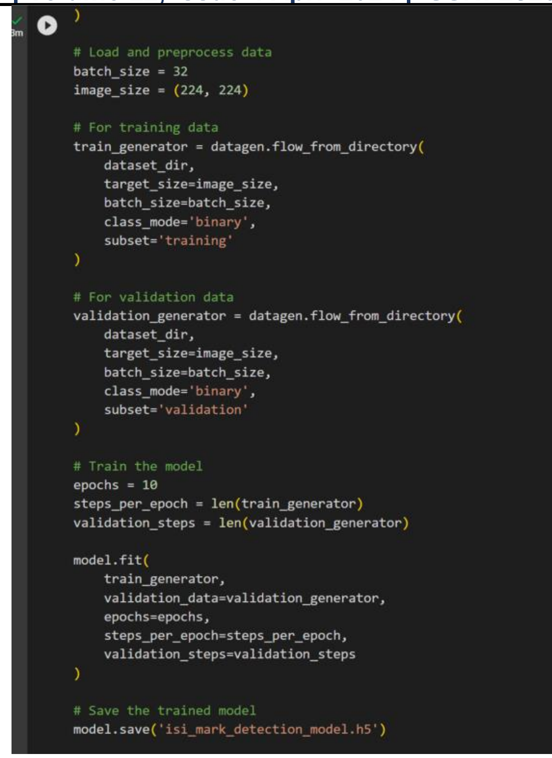
```python
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the path to your dataset directory
dataset_dir = '/content/drive/MyDrive/ISI model training '  # Update with the path to your dataset

# Define model architecture
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Data Augmentation
datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.15
)
```

Fig. 1. Importing Various Libraries of Python for Developing System

uses popular deep learning libraries such as TensorFlow and Keras using Convolutional Neural Network (CNN) models [?]. TensorFlow, [?] [?] [?] an open-source machine learning platform, serves as the backbone for building and training the model, [?]while Keras [?], a high- level neural network API, facilitates construction at the model level

IV. Result and Discussion

The dataset in the '/content/drive/MyDrive/ISI model training' directory is an important part of the analysis. It has a collection of diagrams on ISI symbol recognition. The size and diversity of the dataset is important to ensure that the model is robust and generalizable to a variety of real-world situations. The ImageDataGenerator from Keras helps grow datasets by applying transformations such as rotation, shifting, and flipping. This enhancement process enhances the model's ability to handle variations in image input, and helps improve performance during training. The dataset is split into training and validation sets, which allows the model to recognize patterns from the training data while validating its performance on unseen data This split is facilitated by the validation split parameter of the ImageDataGenerator, which checks if part of the dataset is reserved for validation during training. The code specifies a standard CNN setting for image classification services. Convolutional layers are used to capture hierarchical features in images, while MaxPooling layers reduce spatial dimensions, increase computational efficiency Flatten layer prepares data for insertion into densely connected layers, and leads to a final output layer with a sigmoid contains activation function for binary classification In addition, the model is trained for a specified number of periods, repeated across different data sets and weights are adjusted to minimize binary cross-entropy losses The resulting trained model is stored for use future. Overall, the combination of TensorFlow, Keras, and optimized datasets forms the basis of a robust ISI marker detection model, laying the foundation for accurate and reliable predictions



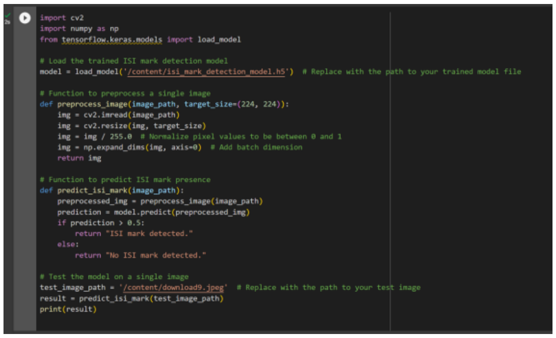
```python
)

# Load and preprocess data
batch_size = 32
image_size = (224, 224)

# For training data
train_generator = datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='training'
)

# For validation data
validation_generator = datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
)

# Train the model
epochs = 10
steps_per_epoch = len(train_generator)
validation_steps = len(validation_generator)

model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=epochs,
    steps_per_epoch=steps_per_epoch,
    validation_steps=validation_steps
)

# Save the trained model
model.save('isi_mark_detection_model.h5')
```

Fig. 2. Loading the Dataset through code in Google Colab

The training process for the ISI mark detection pattern involves several basic steps, beginning with the definition of the pattern structure. The code snippet provided is a Convolutional Neural Network (CNN) using TensorFlow's Sequential API from Keras. The architecture includes convolutional layers for feature extraction, maximum pooling layers for dimension reduction, and densely connected layers for classification. The ImageDataGenerator is used to apply data enhancement to ensure that the sample is generally delivered to different states. This includes recreating pixel values, rotating, rotating, rotating and applying transformations. The data set is divided into training and validation sets, where the model is trained for a specified number of epochs.

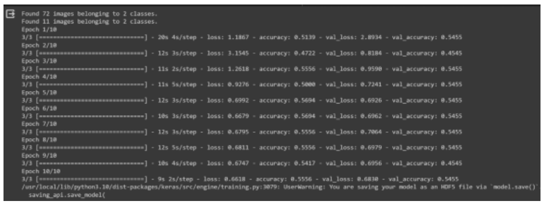
Once the model is trained and validated, it can be used for real-world ISI marker detection. The stored model is loaded using TensorFlow load model function, which allows it to be reused to perform predictions on other models. The statistical procedure involves preprocessing an image, normalizing the pixel values, and expanding the resolution to accommodate the embedded sample size. The model then predicts the presence of an ISI marker, with a probability threshold of 0.5 for binary classification.

These trained models can be incorporated into various applications, such as traffic control systems or safety protocols, to automatically detect the ISI markings on helmets The ability to predict individual images determines the effectiveness of the models in a controlled environment. The code snippets provided are foundational to understand the training and



```python
import cv2
import numpy as np
from tensorflow.keras.models import load_model

# Load the trained ISI mark detection model
model = load_model('/content/isi_mark_detection_model.h5')  # Replace with the path to your trained model file

# Function to preprocess a single image
def preprocess_image(image_path, target_size=(224, 224)):
    img = cv2.imread(image_path)
    img = cv2.resize(img, target_size)
    img = img / 255.0  # Normalize pixel values to be between 0 and 1
    img = np.expand_dims(img, axis=0)  # Add batch dimension
    return img

# Function to predict ISI mark presence
def predict_isi_mark(image_path):
    preprocessed_img = preprocess_image(image_path)
    prediction = model.predict(preprocessed_img)
    if prediction > 0.5:
        return "ISI mark detected."
    else:
        return "No ISI mark detected."

# Test the model on a single image
test_image_path = '/content/download9.jpeg'  # Replace with the path to your test image
result = predict_isi_mark(test_image_path)
print(result)
```

Fig. 3. Training The Dataset and Predicting ISI Mark



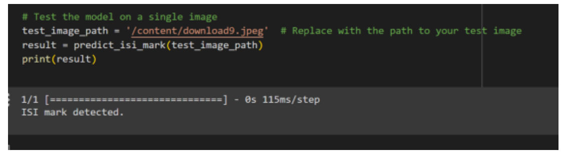
```
Found 72 images belonging to 2 classes.
Found 11 images belonging to 2 classes.
Epoch 1/10
3/3 [==============================] - 20s 4s/step - loss: 1.1867 - accuracy: 0.5139 - val_loss: 2.8934 - val_accuracy: 0.5455
Epoch 2/10
3/3 [==============================] - 12s 3s/step - loss: 3.1545 - accuracy: 0.4722 - val_loss: 0.8184 - val_accuracy: 0.4545
Epoch 3/10
3/3 [==============================] - 11s 2s/step - loss: 1.2618 - accuracy: 0.5556 - val_loss: 0.9590 - val_accuracy: 0.5455
Epoch 4/10
3/3 [==============================] - 11s 5s/step - loss: 0.9276 - accuracy: 0.5000 - val_loss: 0.7241 - val_accuracy: 0.5455
Epoch 5/10
3/3 [==============================] - 12s 3s/step - loss: 0.6992 - accuracy: 0.5694 - val_loss: 0.6926 - val_accuracy: 0.5455
Epoch 6/10
3/3 [==============================] - 10s 3s/step - loss: 0.6679 - accuracy: 0.5694 - val_loss: 0.6962 - val_accuracy: 0.5455
Epoch 7/10
3/3 [==============================] - 12s 3s/step - loss: 0.6795 - accuracy: 0.5556 - val_loss: 0.7064 - val_accuracy: 0.5455
Epoch 8/10
3/3 [==============================] - 12s 5s/step - loss: 0.6811 - accuracy: 0.5556 - val_loss: 0.6979 - val_accuracy: 0.5455
Epoch 9/10
3/3 [==============================] - 10s 4s/step - loss: 0.6747 - accuracy: 0.5417 - val_loss: 0.6956 - val_accuracy: 0.4545
Epoch 10/10
3/3 [==============================] - 9s 2s/step - loss: 0.6618 - accuracy: 0.5556 - val_loss: 0.6830 - val_accuracy: 0.5455
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`
  saving_api.save_model(
```

Fig. 4. Recognizing the Image

deployment pipeline. Further research and development may require further development, real-time programming, and integration with underlying camera feeds that can enhance the practical application of the model to ensure road safety

The real-time data processing using OpenCV together constitutes a valuable development to extend the utility of the ISI marker recognition model beyond still images. OpenCV, a powerful computer vision library, makes it easy to capture video images in real time from the web or other video sources. The following steps illustrate how to obtain real-time ISI marker detection - OpenCV Video Capture (‘cv2.VideoCapture‘) starts acquiring webcams or other video

Fig. 5. Output Showing Sucessfull Detection of ISI Mark on Helmet



```
# Test the model on a single image
test_image_path = '/content/download9.jpeg'  # Replace with the path to your test image
result = predict_isi_mark(test_image_path)
print(result)

1/1 [==============================] - 0s 115ms/step
ISI mark detected.
```

Fig. 6. Final Output

sources. - Each frame in an infinite loop is captured and displayed in the ’ISI Mark Detection’ window. - The frame is previously adapted to the input dimensions of the model, and the model is loaded to obtain the prediction. - The result (whether the ISI symbol is detected or not) is overwritten by the frame using OpenCV‘s ‘cv2.putText‘ function. - The loop continues until the ’q’ key is pressed, at which time video recording is stopped and the window is closed.

This real-time recognition of ISI signals enables a more efficient use of the trained model, which can continuously monitor and detect ISI signals in live video feeds This opens up possibilities for incorporation into channel forcing systems does promote safety and compliance.

V. Conclusion

In conclusion, the developed and implemented ISI signal recognition model makes an important contribution to enhance road safety and compliance The trained model exhibits promising accuracy in ISI signal a it is detected in indirect images, and its real-time integration using OpenCV extends its applicability to dynamic internal events The successful implementation of the model highlights the potential for integration into various policies aimed at improving road safety. The model’s ability to quickly and accurately identify ISI markings on helmets can be applied to traffic managementsystems, law enforcement agencies and security compliance analytics

References

[1] F. Chollet *et al.*, “Building autoencoders in keras,” *The Keras Blog*, vol. 14, 2016.

[2] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O’Reilly Media, Inc.", 2022.

[3] A. Behal and G. Manzoor, “Contiki based routing protocol (rpl) analysis using ipv6 sensor networks for iot with low power and lossy networks,” in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, 2023, pp. 1–3.

[4] T. M. Mitchell, “Machine learning,” 1997.

[5] F. J. J. Joseph, S. Nonsiri, and A. Monsakul, “Keras and tensorflow: A hands-on experience,” *Advanced deep learning for engineers and scientists: A practical approach*, pp. 85–111, 2021.

[6] E. Haghighat and R. Juanes, “Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113552, 2021.

[7] N. K. Manaswi and N. K. Manaswi, “Understanding and working with keras,” *Deep learning with applications using Python: Chatbots and face, object, and speech recognition with TensorFlow and Keras*, pp. 31–43, 2018.

[8] L. O. Chua, *CNN: A paradigm for complexity*. World Scientific, 1998, vol. 31.

[9] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[11] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, “Machine learning in agriculture: A review,” *Sensors*, vol. 18, no. 8, p. 2674, 2018.

[12] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.