







# Conveyor Belt Longitudinal Rip Detection Implementation with Edge AI

Emerson Klippel<sup>1</sup><sup>a</sup>, Ricardo Augusto Rabelo Oliveira<sup>2</sup><sup>b</sup>, Dmitry Maslov<sup>3</sup><sup>c</sup>,  
Andrea Gomes Campos Bianchi<sup>2</sup><sup>d</sup>, Saul Emanuel Delabrida Silva<sup>2</sup><sup>e</sup>  
and Charles Tim Batista Garrocho<sup>2</sup><sup>f</sup>

<sup>1</sup>Vale Company, Parauapebas, Para, Brazil

<sup>2</sup>Computing Department, Federal University of Ouro Preto, Ouro Preto, Minas Gerais, Brazil

<sup>3</sup>TinkerGen, Sseed Studio, Shenzhen, China

Keywords: Deep Neural Network, Device Edge, Rip Detection, Conveyor Belt.

Abstract: Failures in the detection of longitudinal rips on conveyor belts are events considered catastrophic in mining environments due to the financial losses caused and the exposure to safety risks of the maintenance teams. The longitudinal rip detection technologies used today have limitations, being the most reliable systems expensive and complex and the simplest and cheapest systems unreliable. In view of this scenario, we studied the implementation of a longitudinal rip detection solution based on images of the conveyor belt. The images will be collected in real time and inference, rip detection, will be carried out locally using a deep neural network model executed on device edge. The results obtained with the prototype, in controlled field tests, were satisfactory and showed the feasibility of using deep neural network algorithms executed on device edge. These results encourage the development of a complete solution for the detection of defects in conveyor belts considering all the operational conditions found in the mining environment.

## 1 INTRODUCTION

One of the main assets used in iron processing plants are conveyors. Even in medium-sized plants there are more than a dozen kilometers of this equipment installed and in continuous operation. The main component of these conveyors is the vulcanized rubber belts with an internal steel or canvas structure.

The sensitivity to damage by the conveyor belts by piercing and cutting elements is notable, especially in transfer houses where the material transported by the belts is transferred between them. In these places the material being carried and metal scraps can get stuck, acting as blades, causing longitudinal cuts in the conveyor (Hardygóra et al., 1999).


Situations in which longitudinal rips conveyors cause large losses in belts are common, including total losses with impacts related to risk exposure by main-


tenance teams and financial losses due to belt repair and production losses. The Figure 1 shows a belt slot with total loss of the asset.


The most common belt tear sensors found in the iron ore mining environment are electromechanical and electronic. Electromechanical devices are not very reliable since they need a direct interaction between the tear or its effects and the sensor, on the other hand they are inexpensive and easy to maintain. Electronic sensors are more reliable but have a high cost and require the installation of sensor elements in the belt structure, not allowing the interchangeability of belts in case of maintenance demands (Gruenhagen and Parker, 2020).





Figure 1: Image of total loss on conveyor belt by rip.


<sup>a</sup>  <https://orcid.org/0000-0003-0312-3615>

<sup>b</sup>  <https://orcid.org/0000-0001-5167-1523>

<sup>c</sup>  <https://orcid.org/0000-0002-9888-6107>

<sup>d</sup>  <https://orcid.org/0000-0001-7949-1188>

<sup>e</sup>  <https://orcid.org/0000-0002-8961-5313>

<sup>f</sup>  <https://orcid.org/0000-0001-8245-306X>

The objective of our work is to detect longitudinal rips in conveyor belts using images of its surface captured in real time and a model of deep neural network (DNN). This model will make inferences locally using a device edge without the need for centralized processing. This capture of images in real time and execution of the model locally are the main differences of the work when compared to other detection solutions based on image with centralized processing.

The local inference of the model, with all the processing being carried out on the device edge is important taking into account the restrictions of connectivity to the existing network in the case of conveyor belts (Jurdziak et al., 2018).

In general, our work brings an approach based on the AI on Edge paradigm for the solution of an important problem in the iron ore processing industry. More specifically, our solution proposal utilizes the potential of decentralized processing of DNN models running on device edge, eliminating restrictions on lack of connectivity and communication latency, allowing for real-time actions, necessary for the type of solution proposed here.

## 2 THEORETICAL REFERENCE

The evolution of artificial intelligence algorithms, mainly after the development of deep learning algorithms and the hardware optimizations for the execution of these models, has allowed everyday tasks, previously considered feasible only by humans, to be

efficiently performed by machines. This work item addresses, not exhaustively, the theoretical foundation for deep learning and edge AI device.

### 2.1 Deep Neural Network

Modern artificial intelligence models have dozens of layers, thousands of neurons and millions of trainable parameters. This coupled with training techniques using massive volumes of data and specialized hardware for both training activities and model inference led to deep learning algorithms or deep neural network (DNN). These algorithms have become practically standard for systems that involve image classification and detection (Koul et al., 2020).

Deep learning models with convolutional layers are used to detect and classify images. These layers work as filters, extracting characteristics from the images (Krizhevsky et al., 2017). In the implementation of deep learning optimization techniques are used in order to reduce the size of the model and its processing demand, among them we have dropout, parameter sharing and rectified linear units (ReLU) activation functions (Krizhevsky et al., 2017).

An important concept used in DNN is that of knowledge transfer. This technique allows the use of previously trained models from specific datasets, simplifying the final training process. In this case, it is not necessary to carry out the training from scratch, reducing the size of the datasets and consequently the training time and complexity (Krizhevsky et al., 2017).

The implementation of the deep learning algo-

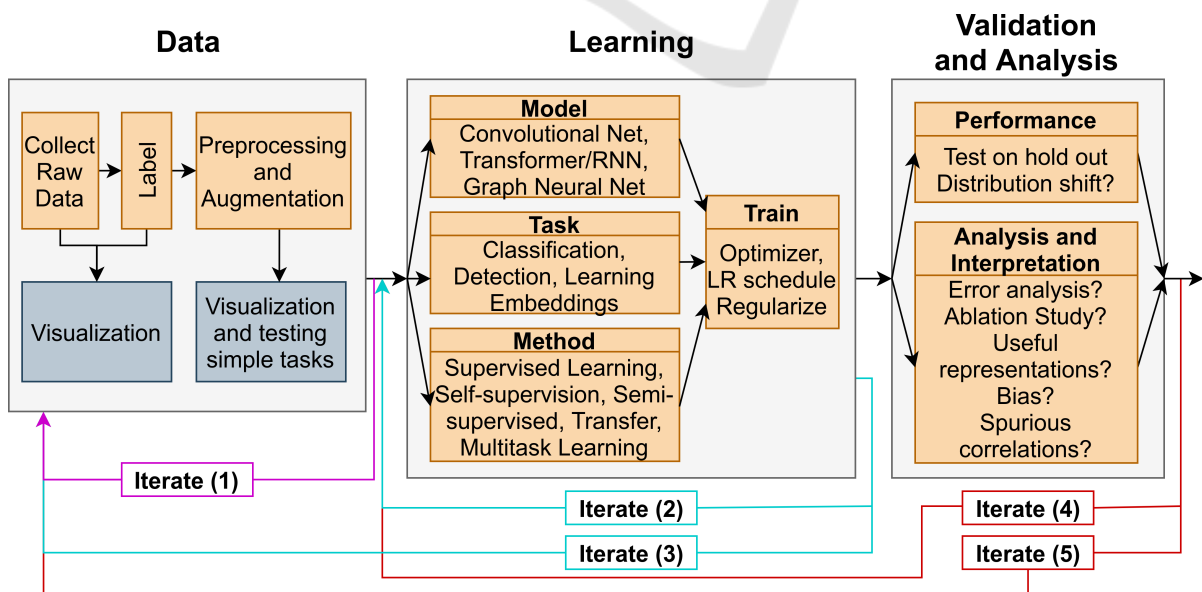


Figure 2: Stages of DNN implementation.

rithms can be divided into three stages as can be seen in Figure 2. The first step consists of processing the data to be used in the training process. In the second stage, the models to be used are defined considering the task to be performed, the training methods, its execution and optimization. In the third stage, performance tests are carried out in a set of analyzes that aim to detect generalization capacity, bias and spurious correlations (Raghu and Schmidt, 2020).

## 2.2 AI on Edge

The edge computing paradigm was born from the current demand for decentralized processing, mainly considering the volume of data produced by the Internet of Things (IoT) devices. At a high level, edge computing can be divided into Device Edge, Enterprise Edge, Far Edge, and Near Edge. We can see these levels in Figure 3. Devices Edges are closer to the real physical world, collecting information from the environment, processing these locally and taking actions that interact directly there (Bertino and Banerjee, 2020).

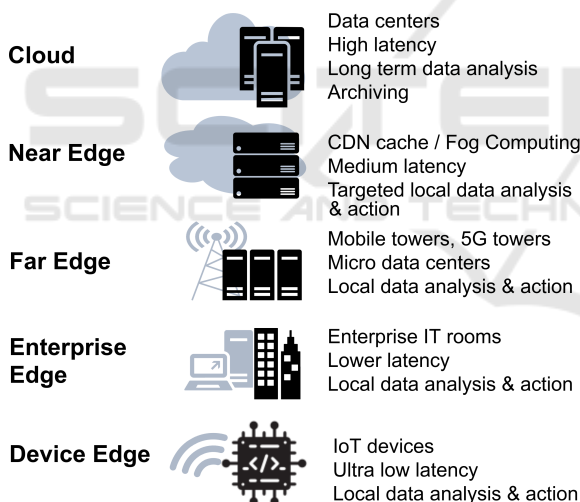


Figure 3: Edge computing classification.

The convergence between AI and device edge creates a new field of studies and application development, called AI on Edge. In these studies, efforts are concentrated on the design of training and inference frameworks, adapting the models and accelerating the hardware for use with AI (Deng et al., 2020b).

The adaptation of models and training and inference frameworks are being carried out by device Edge manufacturers for AI applications. In our studies, we verified the existence of solutions such as the software of the Google Coral platform (LLC, 2020), the nncase compiler (Sunnycase, 2020), and the Open-

Vino toolkit from Intel (Corporation, 2020). The purpose of these solutions is to adapt models trained in the cloud or in hardware with a high processing power for later use in Edge AI, devices with limitations in processing capacity and memory.

Processors for AI on Edge applications are built taking into account energy constraints, memory capacity and processing speed. The optimization of these processors for the execution of AI algorithms concerns the architecture of memory and the parallelism of logical operations. (Deng et al., 2020a).

The generic architecture of a neural network accelerator can be seen in Figure 4 being composed of an array of processing elements (PE), each with a small memory buffer, a global buffer for compensating latency for accessing external memory. The PE process the input and output activation functions, the network weights, and the sum result function. The joint execution of these functionalities by the PE allows the implementation of the convolution, polling, and feedforward processes of the deep neural networks (Deng et al., 2020a).

## 3 RELATED WORK

Techniques with the use of computer vision, computer hearing and deep neural network to detect equipment failures in an industrial environment are proposed by several authors. In this topic we cover some of these applications.

One of the proposals for the detection of longitudinal rip in conveyor belts is the use of computer vision combined with laser light to detect the failure. The belt is illuminated by a laser on its surface opposite the load region, a CMOS matrix captures the image formed by the laser trace. This image goes through filters and distortions in the design line by the laser are extracted and detect the failure. In simulations, the system showed the ability to detect with tears quickly and with accuracy (Xianguo et al., 2018). In another approach, the accuracy of 86.7% was obtained in laboratory tests of the detection system based on the combination of sounds and images. In this system, real-time images obtained from the bottom of the belt are processed with the application of filters, binarization and bit counting, this counting is associated with the existence or not of tears in the image. At the same time the noise produced by the belt is captured close to the camera's installation point, this signal is processed using Mel-Frequency Cepstral Coefficients (MFCC) and Gaussian Mix Model - Universal Background Model (GMM-UBM) to identify possible signatures that can identify a tear. Video and au-

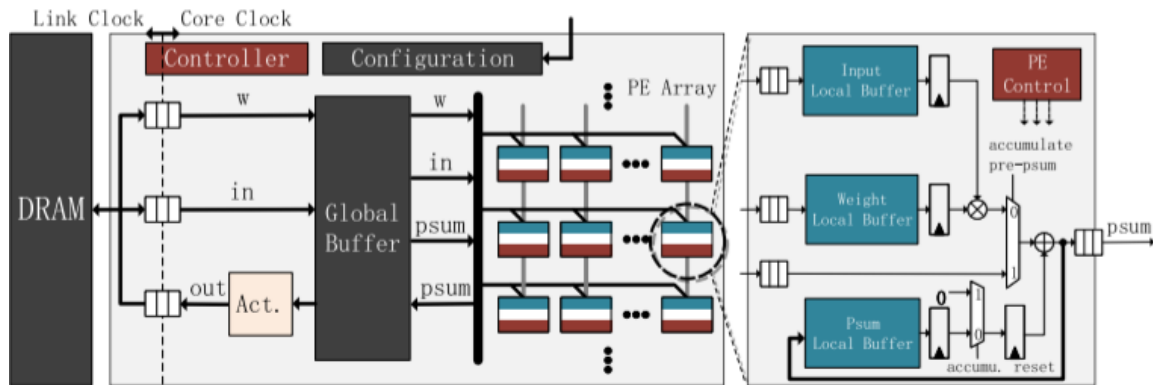


Figure 4: The typical digital architecture of neural accelerators.

Table 1: Device edge comparison.

	<b>Raspberry Pi 3</b>	<b>Jetson Nvidia Nano</b>	<b>SiPEED MAiX BiT</b>
<b>Processor</b>	ARM Cortex A53	Quad-core a57	K210 RISC V
<b>Clock (GHz)</b>	1.2	1.43	0.40
<b>RAM (GB)</b>	1	4	0.008
<b>AI resources</b>	NA	GPU	KPU
<b>OS / Language</b>	Raspian / Python	Ubuntu / SDK JetPack	uPython
<b>GPIO pins</b>	40	40	48
<b>Power rating (W)</b>	15	10	5
<b>Dimensions (mm)</b>	85x56x17	100x80x29	52x39x10
<b>Costs (US\$)</b>	75.00	194.00	21.00

dio processing information is combined to determine the presence or absence of rip in conveyor belt (Hou et al., 2019).

For the detection of dirt in the belt conveyor structures based on images, convolutional neural networks were used with promising results (Santos et al., 2019). ResNet18 and VGG16 architectures were used. These models were trained from 73 photographs of clean and dirty conveyor structures. Data augmentation techniques were used to increase the generability of trained models. Accuracy results of 95.5% were obtained for the VGG16 mode and 81.8 % for ResNet.

study where YOLO (You Only Look Once) and Faster R-CNN (Faster Region Convolutional Neural Network) models were used to detect damage on paved roads and highways obtained satisfactory results with a precision of 93% and F1 (Overall Accuracy) and 84% for YOLO and precision of 75% and F1 of 65% for Faster R-CNN (Majidifard et al., 2020). For the construction of the training dataset, 7,237 images from the Google Street View were used. Each of these images was classified by a human specialist among 9 categories. This study is relevant to our work due to the fact that the image of the damage to the as-

phalt of the street is similar, in aspect, to the damage found in conveyor belts.

## 4 DEEP NEURAL NETWORK IN EDGE AI

For the use of DNN at Edge, the platforms available on the market at the time of the start of the work were evaluated. The one with the best cost-benefit ratio was the MAiX BiT board from the manufacturer SiPEED. The relevant characteristics evaluated for choosing the board, considering the scope of the work, are shown in the Table 1.

Below we discuss the main elements of SiPEED architecture used in the implementation of our work.

### 4.1 SiPEED Architecture

The core of the SiPEED MAiX BiT board is the K210 Kendryte System on Chip (SoC) designed specifically for computer vision and hearing applications, in addition to having an accelerator for convolutional neural

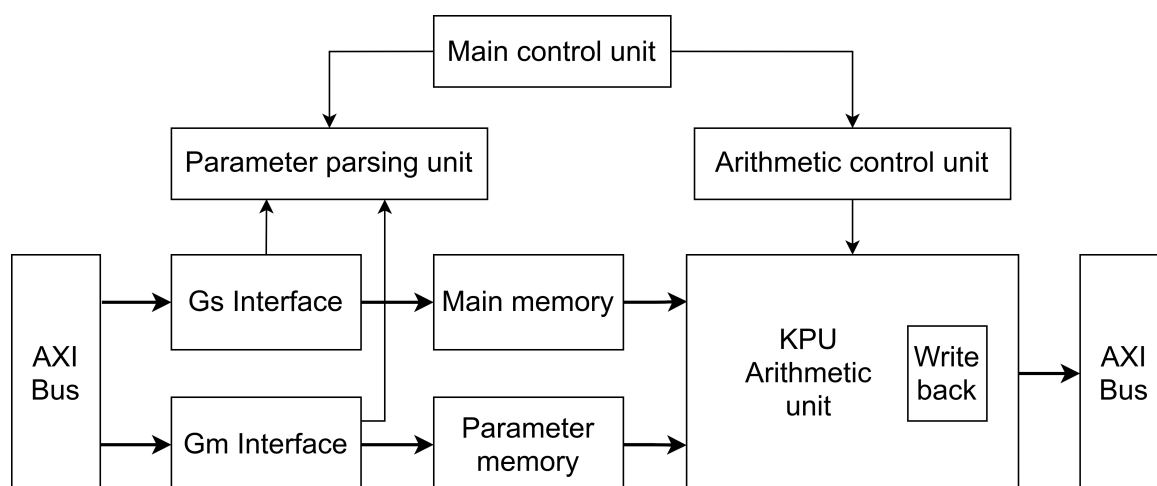


Figure 5: KPU block diagram.

networks. The K210 block diagram is shown in Figure 6.

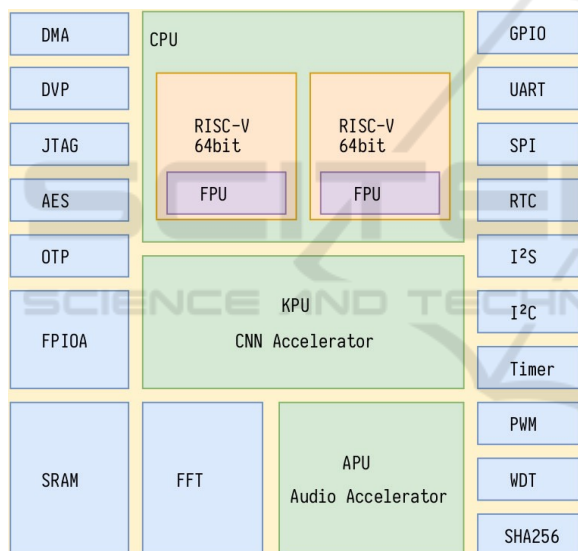


Figure 6: Block diagram of K210.

SiPEED boards can be programmed using C or Micropython programming languages, C SDK and Micropython firmware both have a specific set of libraries for manipulation of convolutional neural networks, computer vision, and sound or voice. When flashed with Micropython firmware, SiPEED boards can be programmed using MaixPy IDE, an integrated development environment derived from OpenMV, that allows connection to the device, code execution, and debugging with visual feedback.

The K210's convolutional neural network accelerator is the Knowledge Processor Unit (KPU) being prepared to perform convolution, batch normal-

ization, activation and pooling operations. The KPU can be interfaced directly with the Digital Video Port (DVP) for real-time applications. Figure 5 shows the K210 Kendryte KPU block diagram.

The KPU supports a wide range of tensor operations used in common network architectures, such as Conv2D, DepthwiseConv2D, MaxPool2D, Relu6, and others(20+ in total). Model compilation to K210 format(.kmodel) is performed using nncase software developed by the K210 manufacturer. The manufacturer of the K210 is the Chinese company Cannan Creative.

## 4.2 Training Framework

The aXeLeRate framework was used in training the deep learning model implemented in Edge AI. aXeLeRate is based on Keras-TensorFlow and consists of a set of scripts optimized to be executed in a jupyter notebook running on the Google Collaboratory platform (Maslov, 2020).

AXeLeRate has a modular structure, allowing users to combine different frontend architectures with a variety of feature extractors, such as MobileNet, NASNetMobile, ResNet, and others. Frontend defines the format of data output by model - in aXeLeRate users can choose between a classifier, YOLOv2 detector, and SegNet-basic semantic segmentation network.

The data in front of the images are preprocessed and fed into the feature extractor part of the network. The resulting feature vectors are used by the network frontend to classify the image, output the bounding boxes or segmentation masks, depending on the type of frontend.

The main feature of aXeLeRate is the automatic

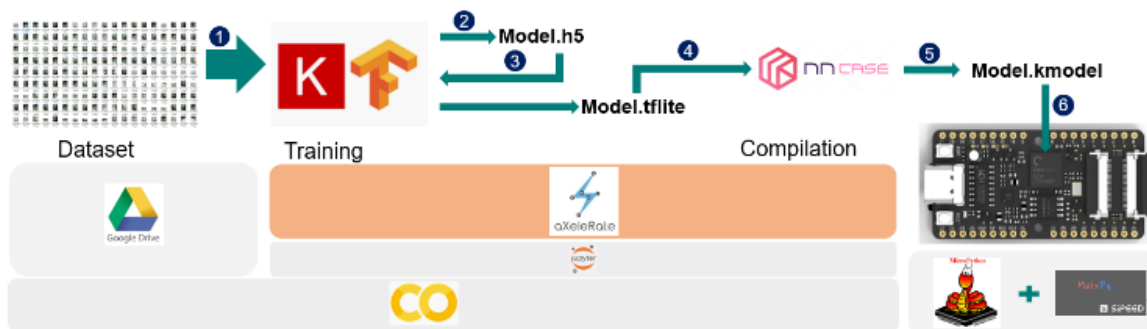


Figure 7: Training and compilation with aXeLeRate.

conversion of trained models to the necessary format for later use on Edge AI devices. The Edge AI devices ecosystem is currently very fragmented, each device requires the model to be converted into its own format in order to accelerate inference. The conversion is done using different tools that are often not compatible with each other. For example, K210 uses nncase converter, Nvidia Movidius chips use OpenVINO toolkit and Google Edge TPU uses a proprietary model compiler. aXeLeRate aims to make the process of model training and conversion to different formats for hardware-accelerated inference more convenient and less time-consuming for the end-user.

The process of using aXeLeRate is shown in Figure 8 with the main steps indicated by the blue circumferences. The first step consists of loading the images from the dataset stored in Google Drive for training in the TensorFlow-Keras framework (indication 1). After training, the model is delivered in .h5 format, for classifiers (indication 2). Next, the .h5 model returns to TensorFlow (indication 3), to be converted to .tflite format (indication 4), and then to be compiled in nncase. The nncase compiler performs the compression, parameterization, and compilation of the model to the .kmodel format (indication 5). The .kmodel model is executed by the KPU from the device's SD card (indication 6).

### 4.3 DNN Selection

In selecting the model, we take into account the models that can be compiled by nncase. Among these models, we selected MobileNet due to its architecture being efficient in terms of fine-grained recognition, accuracy, and low computational cost (Howard et al., 2017). The comparison between the different versions of MobileNet and the fine-grained recognition benchmark, in this case Inception v3, is shown in Table ref tab: tab2. The Stanford Dogs dataset was used to assess this capacity of the compared networks

Table 2: Model Comparison using Stanford Dogs, (Howard et al., 2017).

Model	Top 1 Accuracy	Million Parameters
Inception v3	84%	23.2
1.0 MobileNet-224	83.3%	3.3
0.75 MobileNet-224	81.9%	3.3
1.0 MobileNet-192	81.9%	1.9

## 5 METHODOLOGY

This section describes the training methodology for the deep neural network, building the Edge AI prototype and field tests for the study.

### 5.1 Edge AI Prototype Construction

The prototype was built with the SiPEED board to carry out field tests and capture images of the conveyor belt. A prototype was built from the SiPEED board to obtain photos of the belt and field tests. In the prototype, the SiPEED was protected from the existing dust and moisture near the conveyor belt. The final structure of the prototype is shown in Figure 8.

The set thus built was installed on a tripod to allow correct positioning and mobility between tests. In the enclosure, fault indication LEDs were installed to facilitate the monitoring of the tests. The SiPEED electrical supply was provided by a 5V and 10A power bank installed inside the enclosure.

For the tests, three Python scripts were developed. The first to capture photos in the field with 224x224 resolution and storage on the SD card. The second for testing the model from the validation dataset previously stored on the SD card. The third for damage classification tests in the field with storage of the classified photos with this situation on the SD card.



Figure 8: Edge AI prototype.

### 5.2 Training the DNN

The data set was developed to train the deep learning model. For the dataset 396 photos of the damaged belt (tear) and 396 photos of the normal belt were taken. The damage simulations were carried out by the maintenance team, Figure 9, and pictures of these situations were taken with the belt stopped and in motion.



Figure 9: Edge AI prototype.

The photos were taken with SiPEED itself using the 224x224 resolution appropriate to the MobileNet input format. Examples of these images are shown in Figure 10. The images of each class were divided into training images 360 and 36 verification images.



(a) Without tear. (b) With tear.

Figure 10: Images of conveyor belt.

The 0.75 MobileNet architecture was configured as a classifier, with 224 inputs, 1 standard convolutional layer, 26 depthwise layers followed by batch normalization and ReLU activation function and 2 fully-connected layers with 100 and 50 neurons, and a dropout of 0.5. The training was carried out using aXeRate / Keras-TensorFlow in Google Colaboratory. The training was carried out in 10 epochs and the learning rate adopted was 0.001. The initial weights of the model are loaded considering previous training with the ImageNet dataset.

### 5.3 Experiments

For the analysis of results, it used confusion matrices with that of Figure 11 at the same time that they used the parameters precision (1), recall (2), and overall accuracy F1 (3). Where TP is truly positive, FP is false positive, TN is true negative and FN is a false negative.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{3}$$

		Predict	
		Class 1	Class 2
Real	Class 1	TP	FP
	Class 1	FN	TN

Figure 11: Confusion matrix definition.

For DNN performance tests on the Edge device the same images used in the validation of the training performed by aXelerate are stored on the SD card of the SiPEED board. From these images the inference is made by the model compiled and executed in the

KPU. The classification results are stored in a txt file for further analysis.

In the field tests we used the third script, the tests were conducted on the conveyor belt of a bucket reclaimer in the process of demobilization, in Figure 12 shows the equipment in question.



Figure 12: Bucket reclaimer used for field tests.

A single prototype (one sensor) was positioned close to the belt so that the left side of the belt and half of its center were in the SiPEED’s field of view, Figure 13. The maintenance team simulated cuts in the belt and it was activated so that cuts would pass in front of the prototype. It was defined that each simulated cut would pass in front of the prototype 10 times. For each detection, correct or not, the image would be stored. Timed photos every 10s were also taken by the prototype for future use.

During the positioning of the prototype, care was taken to minimize environmental influences such as shadows and glare, both by natural and artificial light.

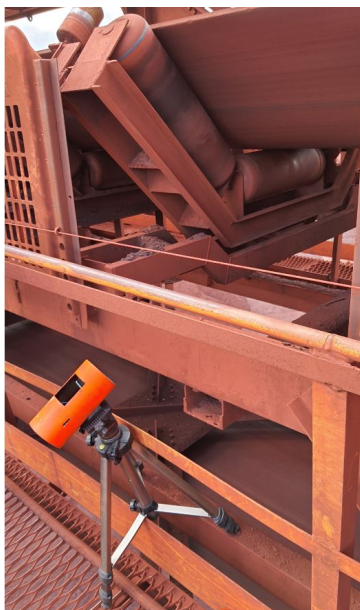


Figure 13: Instalation position of prototype.

## 6 RESULTS

This section presents the results of the field tests with the prototype and the general performance of the deep neural network model. Validation of the model compilation process is also addressed.

### 6.1 Trained Model Results

The entire training process for the MobileNet-224 model was carried out on the aXeLeRate platform using the previously prepared belt tear image dataset. The weights backend used in the training was based on the ImageNet dataset. The training was carried out in 10 times taking 3 minutes in the process and reaching the 94.6% accuracy shown in the Figure 14 and the loss during the training is shown in the Figure 15.

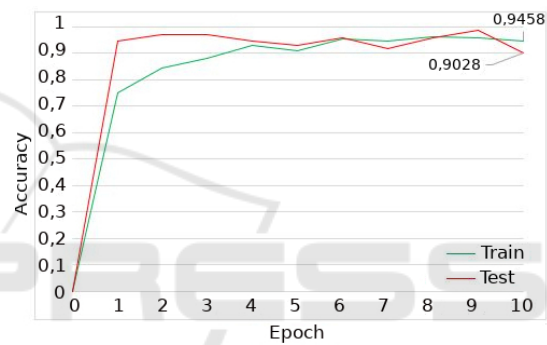


Figure 14: Training graph.

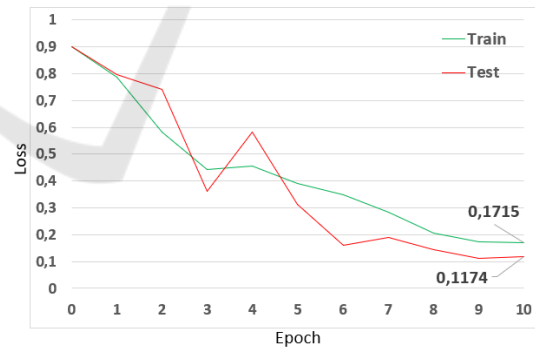


Figure 15: Loss graph.

The model was evaluated using the set of verification images separate from the original dataset. Altogether there were 36 images with tears and 36 without tears. The confusion matrix is shown in Figure 16. The performance indicators can be seen in Table 3.

### 6.2 Performance of AI on Edge

The same set of images used in the training validation of the MobileNet 0.75 neural network model was an-



		Predict	
		Rip	Normal
Real	Rip	36	0
	Normal	1	35

Figure 16: Validation confusion matrix - Google Colab.

Table 3: Model Results in Google Colab.

Indicator	Value
Precision	100%
Recall	97%
F1	99%

alyzed by the SiPEED MAiX BiT board, loaded with that optimized model. The results obtained are shown in Figure17 and Table 4.

		Predict	
		Rip	Normal
Real	Rip	36	0
	Normal	1	35

Figure 17: Matrix confusion model execution in SiPEED.

Table 4: Model Results in SiPEED.

Indicator	Value
Precision	100%
Recall	97%
F1	99%

When we compare the results obtained in the validation of the aXeleRate training process, Table 3, with the results of the model validation in SiPEED, Table 4, they are the same indicating that the compaction process performed by nncase in compiling the model for use by KPU did not cause losses in its indicators.

### 6.3 Performance of Field Tests

The field experiments were carried out in 9 campaigns where the prototype was installed close to the belt so that the simulated tears in the belt were in the sensor’s field of view. The installation location of the belt is shown in Figure18.

In the tests 10 regions with tears and 10 regions without tears were presented for the prototype. Both for tear detection situations and for normal belt situations, the prototype took pictures of the belt. The



Figure 18: Field installation.

tests were conducted under lighting conditions ranging from 800 to 10000 lux. The SiPEED imaging rate was 6 fps during the tests. The 9 experiments totaled 180 exposures with the results presented in the confusion matrix of Figure 19 and the performance of Table 5. Images of the prototype classifying the belt situation are shown in Figure 20. The results obtained in the field tests were satisfactory when compared to the works consulted and in the same line of study indicating the feasibility of using AI on Edge as a solution for detecting longitudinal rip of conveyor belts in industrial environment.

		Predict	
		Rip	Normal
Real	Rip	90	0
	Normal	7	83

Figure 19: Field test confusion matrix.

Table 5: Field Test Results.

Indicator	Value
Precision	100%
Recall	93%
F1	96%

## 7 CONCLUSION

The training process of the deep neural network for the detection of longitudinal rip in conveyor belts, its conversion and compilation for later use in Edge AI device showed satisfactory results. For the same set of validation images, the results of the model classification were the same both when executed in Google

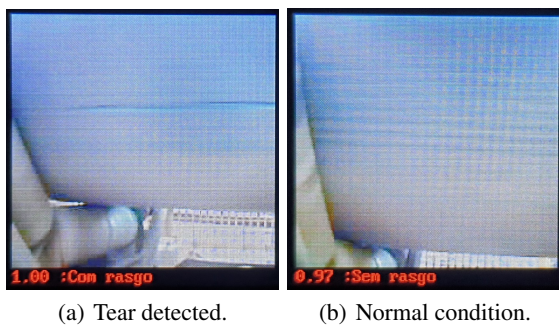


Figure 20: Images of SiPEED prototype in operation.

Collaboratory and executed by SiPEED's KPU.

The results of precision (100%), recall (93%) and total overall accuracy (96%) obtained during the 9 field tests performed were satisfactory and indicate the feasibility of using edge AI with the MobileNet deep learning model for the detection longitudinal rip on belt. With these positive results we understand that other failure modes, with distinct visual characteristics such as misalignment, contamination of the belt return and seam failures can be investigated.

As the objectives of the work were achieved, the process of building 5 more prototypes for definitive installation on 2 conveyor belts and continuous monitoring of their performance was initiated, considering the normal operational conditions of the iron ore beneficiation plant environment.

Continuing the development of the belt failure detection system, new functionalities will be developed, such as automatic verification of the cleaning condition of the optical system lens, detection of the correct positioning of the sensor and detection of failures in the lighting conditions. These improvements are necessary to guarantee the reliability of the solution in the operational conditions verified in the area.

## ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ), the Instituto Tecnológico Vale (ITV) and the Universidade Federal de Ouro Preto (UFOP).

## REFERENCES

Bertino, E. and Banerjee, S. (2020). Artificial intelligence at the edge. *arXiv preprint arXiv:2012.05410*.

- Corporation, I. (2020). Opencv toolkit.
- Deng, L., Li, G., Han, S., Shi, L., and Xie, Y. (2020a). Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532.
- Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., and Zomaya, A. Y. (2020b). Edge intelligence: the confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*.
- Gruenhagen, J. H. and Parker, R. (2020). Factors driving or impeding the diffusion and adoption of innovation in mining: A systematic review of the literature. *Resources Policy*, 65:101540.
- Hardygóra, M., Wachowicz, J., Czaplicka-Kolarz, K., and Markusik, S. (1999). *Conveyor belts*. WNT Warszawa.
- Hou, C., Qiao, T., Qiao, M., Xiong, X., Yang, Y., and Zhang, H. (2019). Research on audio-visual detection method for conveyor belt longitudinal tear. *IEEE Access*, 7:120202–120213.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- Jurdiak, L., Blazej, R., and Bajda, M. (2018). Conveyor belt 4.0. In *International Conference on Intelligent Systems in Production Engineering and Maintenance*, pages 645–654. Springer.
- Koul, A., Ganju, S., and Kasan, M. (2020). Practical deep learning for cloud, mobile, and edge: Real world ai & computer vision projects using python, keras & tensorflow.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.
- LLC, G. (2020). Coral ai.
- Majidifard, H., Jin, P., Adu-Gyamfi, Y., and Buttlar, W. G. (2020). Pavement image datasets: A new benchmark dataset to classify and densify pavement distresses. *Transportation Research Record*, 2674(2):328–339.
- Maslov, D. (2020). Image recognition with k210 boards and arduino ide/micropython.
- Raghu, M. and Schmidt, E. (2020). A survey of deep learning for scientific discovery. *arXiv preprint arXiv:2003.11755*.
- Santos, A. A., Rocha, F. A. S., Azpúrua, H., Reis, A. J. R., and G., G. F. (2019). Automatic system for visual inspection of belt conveyors. *Intelligent Automation Symposium*, pages 1192–1197.
- Sunnycase (2020). Kendrite nncase.
- Xianguo, L., Lifang, S., Zixu, M., Can, Z., and Hangqi, J. (2018). Laser-based on-line machine vision detection for longitudinal rip of conveyor belt. *Optik*, 168:360–369.