Title:

Comparison and Optimisation of CNN Accelerators'

Abstract

The area of computer vision has significantly evolved over the years especially in handling problems in classifying and detecting images. The developments in fast-CNN, fully convolution network (FCN), DL frameworks and algorithms have facilitated more advances in the areas of detecting objects and semantic segmentation in dynamic and complex image data. At the same time image classification and object detection problems face challenges related to need for higher computing systems, speed and performance. To achieve higher accuracy, less energy and performance accelerator boards and optimizer techniques are available that help in reducing resource needs. The research presents a CNN model for image classification and object detection using the coco 128 dataset. These datasets as inputs will train and test the CNN model in different accelerator development boards. The results of Google Coral and Nvidia Jetson Nano board were investigated. The performance evaluation involved the use of different accelerator boards and values of performance benchmarked using training dataset and inference datasets as input. The optimization techniques of Resnet-18 and YOLO are used to evaluate the test datasets on accelerator boards. The image classification results yielded over 90% accuracy. After increasing the hyper parameters the datasets were trained on ResNet-18 algorithm to obtain an accuracy of over 80%. It is noted that more training can help to achieve higher accuracy. The research underlines the use of accelerator boards and optimizer algorithms in overcoming problems in image classification and detection in terms of energy consumption, efficiency and speed.

Acknowledgement

Table of Contents

List of Figures

1 Chapter 1: Introduction

Image classification and object detection has become a significant area for research especially in the area of artificial intelligence (AI). Recent developments in deep learning (DL) frameworks a subset of machine learning (ML) and convolutional neural networks (CNN) have improved performance levels in applications. Improved performance in applications include detecting objects, image-segmentation, human-pose estimation, image detection from video streams, static image classification, and so on. The technology of image lassification has significantly improved and has become essential in the development of computer vision, that has the main features of data pre-processing, feature extraction, object-representation, and in design of classifier. Also, image classification research focuses on these computer vision features. Recent trends also focus on overcoming traditional image feature that leads to poor generalization, portability and ability. Hence to overcome limitations, developing ability in the computer to process images that are similar to biological vision is needed. The developments in artificial neural network (ANN) algorithm supported by mathematical models provided the direction for analysis of image datasets with added learning functions and accelerator elements and optimizer algorithms led to development of new approaches and techniques in image classification and object detection applications.

1.1 Brief overview of CNN algorithm in object detection and recognition of new objects

As DL is a subset of ML which is a subset of AI. DL algorithmic and techniques are involved in different applications under image recognition, hand writing recognition, voice recognition, and so on. CNN is a subset of DLs and used widely in image recognition and object detection applications, particularly images from video streams. CNN algorithm can be trained for recognizing objects or images and can be implemented in embedded devices for inference.

Inference devices are commonly computers, Raspberry pi, Google Coral, Nvidia Jetson Nano, and Intel Neural Compute Stick (NCS). These are also known as accelerator boards used in image segmentation and recognition and object detection applications. The board can embed with existing systems and devices for improving performance of image classification and object detection applications. New objects can be trained on the CNN model to detect accurately and classify images.

1.2 Statement of research problem

The research focuses on the use of accelerator boards to evaluate the performance of image classification and objection using CNN techniques. The CNN model is usually written in Python and the image dataset used is the open-source coco 128 dataset. The CNN model is evaluated using different accelerator boards and the benchmark values determined. The benchmark established is further optimized with optimisation techniques namely ResNet-18 and YOLO algorithms. The optimizations are benchmarked to identify the best performing board, model and the hyper-parameter sets. The investigations are done to understand the board that can provide performance characteristics such as speed, classification accuracy and energy consumption.

1.3 Research Objectives

- Explore the techniques of CNN and evaluation metrics related to performance in the area of image classification and image detection
- Explore CNN models and frameworks from literature and identify gaps in existing research literature
- Develop CNN model for image classification and object detection using the coco 128 dataset available in open source
- Train the model using training dataset

- Identify accelerator boards for implementing inference datasets and embed with the CNN model
- Obtain performance metrics for the accelerator boards used
- Evaluate the performance for the board after implementing optimizer algorithms namely ResNet-18 and YOLO SDG
- Evaluate performance of the board after optimiser to understand the best performing accelerator board

1.4 Significance of research

The area of image classification and object detection using CNN models face challenges in terms of accuracy, speed, resources and energy efficiency. There are also many accelerator boards made available for use in applications with embedded devices. The project makes use of accelerator boards to understand the performance of CNN models in the area of image classification and object detection. The dataset used as inference and the training model will be implemented on each accelerator board. The performance metrics for each board are obtained and he be performing board for use in image classification and objection detection applications is identified. Given these aspects the project is considered significant. The scope is limited to use of accelerator boards in CNN models to evaluate performance in the area of image classification and object detection.

1.5 Structure of the dissertation

The report is developed to have the following sections as chapters.

Chapter 1 provides a high-level introduction to the research topic along with the problem statement, project objectives and significance of this research. Chapter 2 presents the literature review by exploring secondary research sources and studies summarized. The gaps found in literature are presented. Chapter 3 explains the methodology followed in the research along

with dataset and analysis techniques used. Chapter 4 provides the implementation of techniques and results along with discussions. Lastly conclusions are provided in the report.

2 Chapter 2: Background and Literature Review

2.1 Brief on ML

ML models a major sub-set of AI used to build computer models with capability to learn and make independent decisions or predictions using input data. Accuracy of prediction or decision-making is improved through the process of learning with each input dataset. ML model was first developed in 1959 by Arthur Samuel who is considered the author of ML and explains that with ML computers have the capacity for learning new skills irrespective of separate and distinct programming. The learning is done using available datasets, and the ML algorithm supported by mathematical models can support to make prediction or provide support in decision-making. Two major ML types are namely the supervised learning and unsupervised learning.

Supervised learning: This is one type of ML algorithms that make use of labelled datasets used to train the ML model to predict correctly and recognize patterns. Supervised algorithms are provided with labels to learn the relationship between inputs and outputs.

Unsupervised learning: This is another set of learning algorithms in ML that uses of un-annotated data that is not labelled earlier by data analysts/humans or algorithms. The unsupervised learning model will learn using the input data and will not expect values in the absence of values for the given task from the given dataset. The algorithm in the absence of labelled data will be grouped together on the basis of their characteristics. The main focus is the machine will learn to identify patterns and group similar characteristic data without any correct output. Unsupervised learning methods are of two types namely clustering and association. Clustering refers to grouping data using their similarities or differences. The association method will analyse the relationship between the data and the dataset.

Reinforcement learning: The RL learning technique will train the software to make decisions and arrive at optimal results. This technique is based on the train-and-error learning process used by humans to arrive at accurate results. There are three main approaches to RL namely value-based, policy-based based and model-based.

2.2 Basics of CNN

CNN is based on DL architecture is widely used in image classification problems in many applications and is an interesting area for research. CNN models make use of different network layers known as convolutional networks, activation networks and fully connected- network layers. Each of these layers has a different structure and associated mathematical operations. The layer or different layers are chosen in combination according to the image classification problem (Alzubaidi et al., 2021). According to Oh, Choi and Kim (2020), CNN models are used to classify images and CNNs provide high performance. For instance, images with real-world reflection or photographs having a high correlation with the surrounding pixels can be evaluated using CNN. CNN models have the ability to maintain correlation data directly to result in better performance. Better performance is achieved in CNN by the stacking of two layers namely convolution and pooling.

In images, convolution layer will find data that is hidden from surrounding pixels through the use of linear combination. In this method feature map size is reduced by the pooling layer. Pooling layer will reduce resources required for the model to learn and avoid overfitting. By repeated application of convolution and pooling layers, classification result is achieved by the connected layer. The loss of any noted between the acquired and actual labels is used to train the model through gradient descent or optimizer techniques (Zafar et al., 2022). Stankovic and Mandic (2023) explain that DNN and CNN are models that are the de facto standards for analysing large volumes of signals in images. There are many methods related to the CNN

model in its operation. Basically, CNN operations include pooling, padding and different ways of reducing dimensionality.

2.2.1 Pooling

The pooling operation in CNN will down sample the feature map of output from convolution layer. Pooling will maintain the most relevant information while decreasing the input spatial size leading to a reduction of scope of the feature maps. This is an important layer because the parameter needed for learning is decreased resulting in a reduction in computing on the network. The pooling layer will summarise all features found in the areas on the image of feature map. Hence, subsequent operations are done only on features that are summarised against the features with the precise position in the convolution layer. In this manner, the CNN model is made strong (Zafar et al., 2022). The pooling layer is of three types, namely max-pooling, average pooling and global pooling.

Max-pooling operations will select the maximum elements from the filtered area in the feature map. Output in max pooling is the feature map that contains the major or noticeable features of the feature map. Average-pooling will compute average of all elements in the filtered area in the feature map. This computation will provide average features present in feature map. Lastly, global pooling will minimise every channel in the feature map to some single value (Akhtar and Ragavendran, 2020). All these pooling operations can be performed using Keras in the Python language.

2.2.2 Padding

Padding process is used to add extra pixels on the input image prior to the application of convolution. The extra pixels will provide protection to the boundary around the image to support the network to retain more spatial information. Padding is applied on all sides of the input image in a symmetric fashion. Commonly, zero padding is one strategy where a bunch

of zeros are added around the input feature map. Other padding types include election padding and replicating padding. However, the chosen strategy must support the task at hand. Padding is important because it will help preserve the spatial dimensions of feature maps throughout the CNN architecture (Shyam, 2021).

Padding the boundaries are extended in the input and further ensures each pixel will receive equal treatment during convolution. Equal treatment is important to capture fine-grained information, maintain information related to the position and avoid information loss near the image borders. Padding when applied using some strategy will help CNNs balance feature extraction, spatial preservation and downsampling. The convolution operations are equally made for every pixel (Taye, 2023).

2.2.3 Dimensionality reduction

This technique will minimise features in data while retaining all the important information to the maximum extent possible. Usually, data having a greater number of features or variables are considered high-dimensional data in ML. contrarily in dimensionality reduction when the number of features is reduced the model can deteriorate when dimensions are reduced and hence this is a problem. However, in large image processing problems when the number of features is increased, achieving a good solution can be difficult. Further high dimensional data has other challenges like over fitting where model will fit with training data closely and will not yield the right results on new data. Dimensionality reduction can prevent these problems and improve generalisation performance. Dimensionality reduction has the approaches of feature selection and feature extraction (Zebari et al., 2020). Dimensionality reduction uses the methods of Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Generalised Discriminant Analysis (GDA).

Classification of images has been of significant interest in ML research. Image classification has the main objective of mapping the input data from image with one output class. For the image X as input, this is mapped to Y output-class. For example, a refined classifier can produce an output as a 'car' for a given car image. CNNs have many layers namely the convolutional layers, down-sampling layer and activation layers. These layers operate with some pre-determined function on the input data. Peng et al., (2020) explain that convolution layers extract features that are low-level namely edges, and other layers extract semantic features in the image input. Using a collection of input images, and using the process known as backpropagation. Using the input image data the CNN model learns kernel weights and biases. The values from kernel weights and biases are known as parameters that are used to summarise important features found in images. The kernel weights perform an element-wise dot product by sliding across an input image to provide results immediately. The results are summed together with the learned bias value (Dhruv and Naskar, 2020). A generic architecture of CNN is provided in the Figure 1.

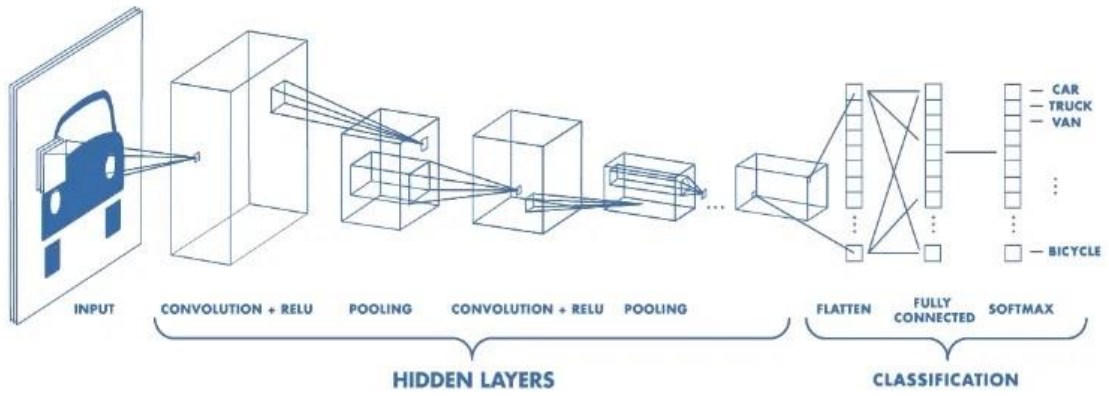

Figure 1: Basic CNN architecture (Mishra, 2020)

Subsequently, based on the input image, each neuron will obtain an output information. These outputs are known as activation maps. The inputs in CNN are down-sampled using pooling.

Here, the parameters are decreased to solve over-fitting problem. Further, the activation functions in CNN introduces nonlinearity. Nonlinearity refers to the model will be able to learn complex information from input datasets. For example, Softmax is an activation function that performs normalization on logits or un-scaled scalar values to produce output class with a score of 1 as the sum (Marcu and Grava, 2021). Hence in comparison, image classification models are over-parameterized and will lead to failure in taking advantage of inherent properties from image data. Whereas, CNN createsrepresentations that are spatially aware using different tacked layers of computation.

2.3 CNN functions

2.3.1 Sigmoid or Logistic Activation Function

The sigmoid function is a fundamental component of ANN and is important in ML applications. This is a mathematical function to map input values between 0 and 1 to make binary classification and is also used in logistic regression problems. In ANNs, sigmoid function is used as activation function and in feed-forward neural networks(NN) where the sigmoid-function is applied to each neuron output to allow the network to introduce non-linearity in the model. Here, non-linearity is important as the neural network is allowed to learn more complex decision boundaries thus improving the performance on specific tasks. This function is also used in DL frameworks such as TensorFlow and PyTorch.

2.3.2 Tanh (Hyperbolic tangent activation function)

Tanh is another activation function having a centre at 0 and values ranging between –1 and +1. This function is used in ANN in hidden layers to transform the input values to produce an out between -1 and 1. This function incorporates non-linearity on the network so that the output function will not be linear-function of input data. This function allows the network will learn more complex and nonlinear relationships between the input and output data.

2.3.3 ReLU (Rectified Linear Unit) function

The ReLU function is used to solve the vanishing gradient issue in the DL model through the introduction of the non-linearity property. The ReLU function is used in NN to speed up training. Like the other non-linear functions, the ReLU function will help in the non-linear transformation of data.

2.4 Evaluation Metrics

In image classification, DL techniques namely classification, segmentation, and object detection are used frequently. These techniques make use of evaluation metrics that are important to refine hyperparameters and features. Classification metrics are used to classify the image. For example, binary classification will classify the object in the image. Binary classification is used in healthcare applications to detect tumours, cancer, etc. Metrics are important in classification models to provide accuracy and make the right decisions (Ferrer, 2022).

Vujovic (2021) explains classification metrics will predict class labels using input data. The binary classification technique has two classes of output. For example, the spam classification of an email is either spam or not spam. This can also be represented as positive and negative or as 1 and 0 for spam and not spam respectively. Classification metrics are used to measure classification performance. Some common metrics are Accuracy, Confusion Matrix, Log Loss, AUC-ROC and Precision-Recall.

Accuracy: The accuracy metric will measure if the classifier will predict correctly. Accuracy is calculated using the values of number of correct predictions divided by the total number of predictions, given by the formula,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Here, TP is true positive, TN is true negative, FP is false positive and FN is false negative.

Confusion matrix: This matrix is used in ML classification problems to measure output performance. The output can be of two or more classes in combination to provide values based on prediction and actual measure. A confusion matrix will explain the performance of classification of the model for the given test dataset for the known true values shown in the Figure 2,

True positive in the matrix implies positive predicted value and true. For example, in the image the predicted value is 'car' and the image is the image of a car.

A true negative implies, a negative value is predicted and it is true. For example, an image can be predicted as a car, but actually, it is some other object.

False positive implies a positive prediction and it is false. Lastly false negative implies the prediction is negative and the actual result is false.

Some of the metrics in the confusion matrix are precision, Recall or Sensitivity, F1 Score, Receiver Operator Characteristic (ROC) and Area Under Curve (AUC)(Demir, 2022, p.317–351).

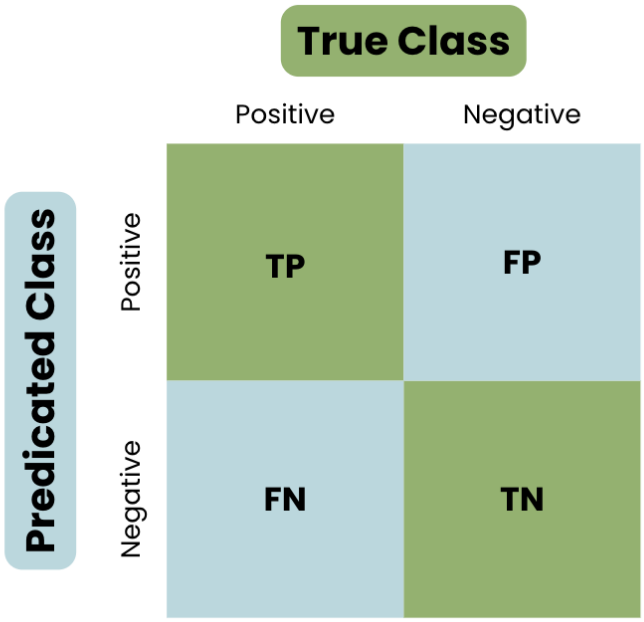

Figure 2: Basic Confusion matrix (Ahmed, 2023)

Precision describes the correctly predicted cases that are positive. The precision metric is useful in cases where the concern for FP is higher than FN.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall (Sensitivity) metric will explain the actual positive cases predicted correctly in the model. This metric is used when FN is higher than FP. Recall provides the value obtained as true positives divided by the total of actual positive values for a label.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Specificity is another metric needed to correctly predict positive class in images. Specificity will explain the negative classes that are predicted and correctly classified.

$$Specificity = \frac{Truenegative}{Falsepositive + Truenegative}$$

The F1 score provides the combined idea to include both the precision-metric and recall-metric. F1 score is maximum both precision and recall are equal.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1-score can be effective when both FP and FN are costly, TN is negative and when adding more data will not change the outcome.

ROC is another measure based on a probability curve or plot showing the true positive rate (TPR) versus the false positive rate (FPR) for different values that meet a threshold and separate signal from noise. AUC measure will distinguish the classifier between classes. Both these measures are graphs plotted using values obtained with earlier metrics.

Another important metric is log-loss or cross-entropy loss to evaluate performance of the classification problem (Demir, 2022, pp.317–351).

The above metrics are available as libraries in Python code with the sklearn metrics library. Further to classification metrics, the metrics for object detection are provided.

Intersection over Union (IOU) provides difference between the prediction bounding-box and the ground truth bounding-box.

$$IOU = \frac{AreaofOverlap}{AreaofUnion}$$

IOU is illustrated in the Figure 3,

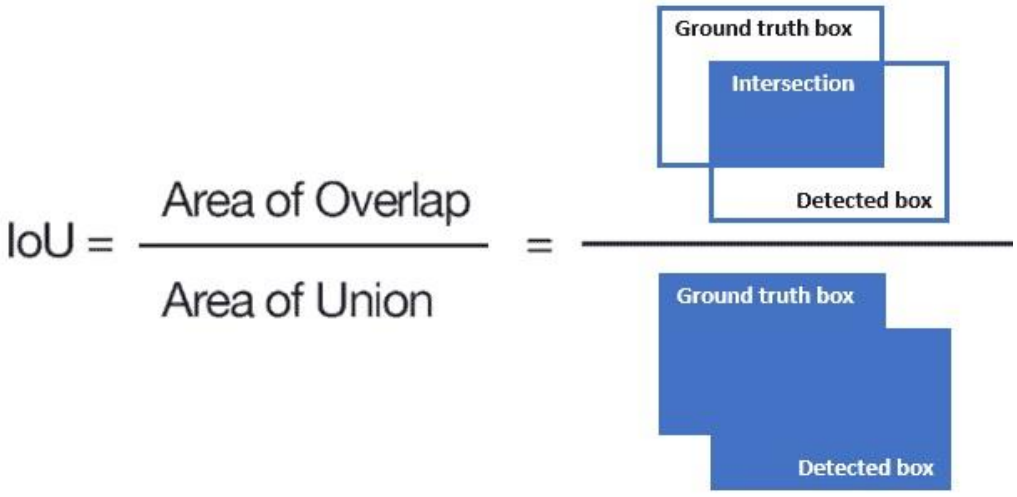

Figure 3: Illustration for intersection over union (Simic, 2024)

mAP is another object detection metric known as mean average precision (mAP). First, the measure namely average-precision (AP) is obtained using area of the precision recall curve. mAP provides the average of calculated AP of a total number of classes to arrive at an accurate value from a group of object detections. The value is then compared with ground-truth annotations of objects in data.

2.5 Datasets

2.5.1 COCO dataset

The data source used is the COCO (Common Objects in Context) dataset available in open source. This dataset is widely used in ML projects. This dataset was created to support advanced development in image recognition. The COCO dataset is a collection of high quality visual images for benchmarking algorithms to evaluate performance of real time object detection and in computer vision advanced neural networks. Some unique features of coco dataset in ludes object segmentation, detailed instance annotation, recognition with 80 object categories and 91 stuff categories having unique contextual information (Meel, 2021).

2.5.2 COCO128

The coco128 dataset is a subset of the 128 images of the coco dataset (Singh et al., 2024). The coco128 dataset is reused as a training dataset for validation and testing and to ensure the training dataset works correctly and can over-fit the small dataset. The Coco128 dataset is commonly used for the first time to test out the CNN models. This dataset provides has visual images in high quality for processing in computer vision and for use in state of-art neural network applications. The coco 128 dataset is used to benchmark algorithms for comparing performance in real-time object detection (Li, Yan and Du, 2022).

Hyper-parameters or variables in the coco 128 dataset: Hyper-parameters refer to variables used to determine the structure of the network or to identify the number of hidden units along with variables used to train the network. Hyper-parameters are set before training or before optimising weights and biases (Hossain and Timmer, 2021). The hyperparameters in the dataset will include,

- Number of hidden layers – implies the number of layers available between the input and output layer. Regularisation technique improves accuracy. Underfitting is a result of a small number of units.
- Dropout – This is a regularisation technique used to avoid over-fitting or to increase the accuracy of validation. A large network will likely provide better performance because the model will learn independent representations.
- Network weigh initialization – Weights are dependent on the activation function used in each layer. The activation function will introduce non-linearity to the models allowing DL models to learn the boundaries in non-linear prediction.
- The sigmoid function is used in the output layer to make binary predictions.
- Softmax used in the output layer will provide multi-class predictions in the data.

Hyperparameters for the training algorithm are: learning-rate, momentum, number of epochs and batch-size.

2.5.3 Large scale visual recognition challenge (LSVRC)

The ImageNet LSVRC (ILSVRC) dataset is used in computer vision to support research in image classification, object-detection and to understand visual images. This dataset has millions of images under thousands of different categories and classes. The images in the dataset are labelled. This dataset is widely used in DL algorithms, CNNs that are trained and benchmarked on subsets of the dataset. For instance, the CNN models such as ResNet, AlexNet, and many others were developed and evaluated using this dataset. ILSVRC dataset is available openly and used by researchers because the dataset is split as training, validation and test datasets with labels. Using this training set the CNN model can be evaluated for performance of trained models (Mandeep, 2024).

2.6 CNN Models and Frameworks

The commonly available CNN models are explained in brief.

2.6.1 AlexNet

AlexNet is a CNN model having 8 layers. This model has the ability to process pretrained state of the network that has over a million images from ImageNet database. This pretrained network has the ability to can images as 1000 object-categories namely animals, mice, pencil, etc. This network provides rich features for a range of images. AlexNet network can be trained on image datasets and the function needs the DL Toolbox model.

2.6.2 ResNet

The ResNet is another CNN architecture introduced by Microsoft Research. When the number of layers in CNN architecture increases it was noted the error rate was high due to fading

gradient. The problem of fading gradient is solved by the ResNet architecture using the concept of residual blocks and skip connections. The skip connection will connect activations to a layer and more layers and skip some layers in between. This method forms the residual block, and the ResNet network will stack the residual blocks together and overcome the limitations found in traditional CNN architectures.

2.6.3 GoogleNet

This network was proposed by Google Research in 2014 to address image classification challenges. The network provided a significant reduction in error rates compared to AlexNet. GoogleNet architecture involves different methods such as 1x1 convolution and global average pooling. These methods enable the creation of deeper architecture. Basically, the 1x1 convolution will decrease the numbers of weight and bias in architecture. This reduction in parameters results in an increase in the depth of the architecture. This architecture is made up of 22 layers and trained on the ImageNet database to classify objects in over 1000 categories.

2.6.4 MobileNet

This network architecture is used in CNN models for mobile vision applications, the network is not resource-intensive. Mobilenet is commonly used in real world applications like object detection, face attributes, image classification and localization. This network is embedded in vision applications and the work is based on depth-wise separable convolutions to build lightweight DNNs to have low latency.

2.6.5 YOLO

You only look once (YOLO) is another type of accelerator to optimize filter switching activity. CNN models make use of YOLO to obtain good performance but YOLO needs large computing and memory bandwidth. YOLO algorithm is popularly used in with real-time object-detection

capacities with high performance. Two optimizers are available for YOLO namely the SGD and Adam.

SDG: The SDG is an optimization algorithm used in ML applications. This algorithm will update the model parameters found in the directions of negative-gradient of loss-function with respect to the parameters. This algorithm will reduce the loss function in ML and hence is considered an optimizer. The model parameters in this algorithm are updated iteratively in small batches of data instead of the entire dataset. This works efficiently for tasks involving complex image structures, but SDG is more effective with smaller datasets.

Adam: The Adam optimization algorithm for SDG is used to train DL models. This is a variant of SDG and combines momentum ideas and adaptive learning rates. The adaptive learning rates are maintained for each parameter where the moving average of past gradients decays exponentially. Adam supports improving performance significantly and combines with LAWN to act as an optimizer in image classification problems.

2.6.6 Frameworks

TensorFlow: The TensorFlow framework developed by Google Research is used in ML, DL and other workloads involving statistical and predictive analytics. This is an open-source framework and platform for ML to includes Python and Java libraries and tools. The framework is designed to train ML and DL models on data. This framework is widely used image-recognition applications.

Pytorch: Pytoch is another DL framework, available in open-source and is a flexible and easy-to-use application. Pytorch is enabled using Python programming and used in developing ML models and in ML applications involving image-recognition and natural language processing (NLP). Pytorch in Python programming is available from the Torch library and is used in computer vision applications.

2.7 CNN for Image Classification Problems

Multiple studies were identified in the literature that explain CNN algorithms for solving image classification problems. Sun et al., (2020) stated that difficulties exist for users in designing a CNN architecture to solve the problem of image-classification. To overcome this limitation, an automatic CNN architecture based on a genetic algorithm is proposed to handle image classification tasks. The automatic characteristic implies the user does not require domain knowledge while using the proposed algorithm but can be used to design a promising architecture for CNN based on the input. The developed algorithm was evaluated with other state of the art peer competitors and used image benchmark image classification datasets. This experimental study indicated the proposed algorithm performs better compared to available algorithms in CNN architecture related to a number of parameters, accuracy and compute resources. However, the proposed algorithm must be further tested to provide accurate performance results with different image inputs from different locations.

Wang, Fan and Wang (2020) provided a comparison of ML and DL algorithms for solving image classification problems using SVM and CNN as examples. The results of this study indicate that SVM provided 88% accuracy while CNN provided 98% accuracy while using the Mnist dataset. The comparison was further made using the COREL1000 dataset and found that the accuracy was 0.86 and 0.83 for SVM and CNN algorithms. The article presents an experimental study to understand DL and traditional ML algorithms and finds that the DL framework has the capacity for higher accuracy even in large dataset samples.

Chen et al., (2021) to understand CNNs provided a detailed review of CNN development and state-of-the-art architecture. The review basically takes into account, structure of ANNs, the basic CNN layer, the classic predecessor model and algorithms. The different image classification methods are also compared and presented in article, summarised and discussed. The article provides a related mathematical model for each of the reviewed network

architectures and models. The article is more theoretical in nature, however, the problems related to efficiency and accuracy found in CNN models were not analysed in detail.

Qin et al., (2020) proposed a classification method for biological images to overcome the drawbacks in CNN when classifying images. The drawbacks considered in the study are improving classification accuracy and making a lightweight network. The classification method made use of fixed-size images as input and the images had large sizes. The method also makes use of an Inverted Residual Block module to reduce computation costs and parameters. The results demonstrate the proposed method shows better performance in image classification with a reduction in network parameters and compute costs. However, the method proposed is experimental and results must be further verified with more image datasets.

Bakhshi, Chalup and Noman, (2020) present a genetic algorithm to optimise the CNN model and to overcome drawbacks on the accuracy, compute cost and performance. The authors used CIFAR10, CIFAR100 and SVHN datasets to experiment and investigate the automatic CNN architecture to reduce cost automatically. The results presented from this experiment indicate that GA evolved CNN model is competitive with other best existing models. However, this claim made in the study must be further verified with different sets of image datasets and from different locations. Valarmathi et al. (2021) made use of the KNN classifier in the horticultural field to identify plant attributes. The framework presented uses content from the picture and a classifier with naturally regulated approaches to determine plant species when the picture is given as input. This plant classification algorithm is a CNN model and will determine whether the leaf is agricultural, herbal or poisonous. However, the idea presented works well for static images, more testing will be needed using camera and sensor devices in the real world.

From the reviewed studies it is noted that image classification using ML algorithms, the DL framework with CNN model provides better performance and computational cost. The studies also highlight the need for more efficiency in terms of classification accuracy and performance.

2.8 Related work on CNN models and applications

Dhillon and Verma (2020) provided a detailed review of different deep architecture models to highlight the characteristics of the particular model. Multiple models of CNN namely LeNet model, AlexNet, ZFNet, GoogleNet, ResNet, ResNeXt, DenseNet, VGGNet, Xception, and PNAS/ENAS are compared and reviewed for detecting wild animal, small arm and human beings from the given input image dataset. Multiple datasets were used and the results were summarised with variations in accuracy and performance.

Du, Zhang and Wang (2020) present a review of wo-stage detection algorithms to explain the working principles of Fast R-CNN, R-FCN, FPN and Cascade R-CNN. The article provides an analysis of the similarities and differences between these algorithms. The performance of Faster R-CNN, R-FCN, FPN and Cascade R-CNN two-stage detection algorithms for its effectiveness by using HSRC2016 ship database set. The results indicate that Cascade R-CNN was highly effective in object detection and the R-CNN algorithm was less effective. Also, FPN and FCN algorithms were less effective in comparison to the Cascade R-CNN algorithm because this algorithm uses a full CNN structure. However, to enhance Faster R-CNN algorithm to be effective efficient feature extraction is needed. This study provides an overview of two-stage object detection.

Sultana, Sufian and Dutta (2020) provided reviews of image-detection models developed using CNN. The review includes both the categories of one-stage approach and algorithms with two-stage approach for image-detection. The study provides a comparison of R-CNN, YOLO and RefineDet (one-stage detector). The architecture is described for these models along with a comparison based on qualitative reviews. Such studies are based on secondary reviews and

comparisons using existing experimental investigations. Studies related to CNN optimiser and accelerator were identified from the literature.

In one study related to brain tumour detection, Hafiz et al., (2023) used the Br35h dataset for brain tumour detection to compare and contrast different CNN models. The models include LeNet, AlexNet, VGG16, VGG19 and ResNet50. The CNN performance was fine-tuned using optimisers namely adaptive movement estimation (Adam), stochastic gradient descent (SGD) and root mean square propagation (RMSProp). The study determined values for accuracy, misclassification rate, sensitivity, specificity, negative and positive prediction values, F1 score and false omission rate (FOR). This metric value was obtained to evaluate the accuracy of CNN architectures when using three optimizers. The results indicate that the SDG optimiser performed better than the other optimisers providing an accuracy of 98.79% and high values for the other metrics. This study though experimental in nature emphasises the need for optimiser in improving the performance of object detection and classification.

Studies highlight the use of optimises in object detection in the area of healthcare. The study by Sharma, Mehra and Kumar, (2020) demonstrates the use of an optimiser and a pooling strategy to detect breast cancer from histopathological images with good performance. Chowdhury, Dasgupta and Nanda, (2021) present the method to show the use of optimisers on different layers of CNN to efficiently detect pneumonia and support physicians in treatment. The study was demonstrated using the chest X-ray image datasets from Kaggle. The SDG optimiser was used to obtain a high accuracy of over 90% with good performance. Gulakala, Markert and Stoffel, (2023) explain that CNN models can be used to determine infected lungs and noninfected lungs in patients. However, such models require large training datasets to ensure high levels of accuracy and performance. In requirements where accuracy must be high, the role of optimisers is significant to ensure classification accuracy. Zhang et al., (2020) presented the use of a CNN accelerator based on FPGA. The accelerator receives signals for

configuration from an ARM to complete the different calculations. Using a combination of CNN and pooling operations the calculation performance was improved. This model was implemented on Xilinx ZCU102 FPGA for YOLOv2 and YOLOv2 tiny models on COCO datasets. The study explains peak performance was obtained at 300 MHz clock frequency.

Kim et al., (2020) proposed a low-power face recognition CNN model with high efficiency on mobile devices with promising results. In another study, Srivastava and Sarawadekar (2020) explain the need for GPU for increasing computing performance. The authors present a pipeline architecture of Depth-wise Separable Convolution based on the layers of activation and pooling as a single-layer CNN. The implementation was done on Xilinx 7 series FPGA to work on a clock-period of 40 nanoseconds. This architecture is important as it provides the foundation for developing an integrated system of CNN accelerators to implement in FPGAs of different configurations. This architecture provides an integrated system design of CNN accelerators to improve performance and support users with FPGA-based accelerators for CNN.

2.9 Summary and Gaps in Literature

In summary, the literature reviews provide explanations of key concepts, with brief coverage of DL architecture and the working of the CNN architecture model. The different DL-related algorithms were also briefly summarised along with the details of CNN operations and techniques. The use of different algorithms that complement the CNN model is discussed in the area of object detection and image classification. The evaluation metrics are provided briefly for their importance in image classification and object detection. The classification algorithms and applications found in studies are provided in the review. The literature on CNN models and applications highlights studies that demonstrate the importance of optimisers and accelerators to improve CNN performance. Some of the challenges identified in the literature include,

- Large datasets are needed to train the model.
- Need for high computing and memory requirements
- The model tends to become slower during the training phase.
- Finding a balance between a smaller number of parameters and performance is a challenge.
- When there is a shortage of labelled data, this impacts the image classification outcomes.

Gaps identified in the literature are,

- Not many studies could be found that highlight the need for reduced training datasets, in most of the studies large training datasets are recommended to achieve accuracy performance.
- Studies highlight the challenges related to data labelling as it can be time-consuming and costly. This is particularly noted when the CNN model is trained from scratch.
- Not many studies were identified that highlight the challenges found in the CNN model.
- Studies that compare different accelerator boards could not be identified immediately in the literature. This project shall address this gap.

3 Chapter 3: Methodology and Implementation

3.1 Setting up Nvidia Jetson Nano

The Jetson Nano developer-kit is open source and is used in inferencing and training DNN networks for object-detection, semantic-segmentation and image-classification. This developer kit involves an AI computer system (chip) to support developers or students to build applications and so on. An illustration with interfaces is provided in the Figure 4 below.

The kit has the ability to capture and process real-time camera feeds and video streams and obtain the dataset using an API available in Python. As a first step, the developer kit is charged using the micro-USB port. A computer system or PC with internet connectivity is needed to read-write on SD cards using a built-in SD card slot. The developer kit with SD image location is found on the computer, based on the operating environment.

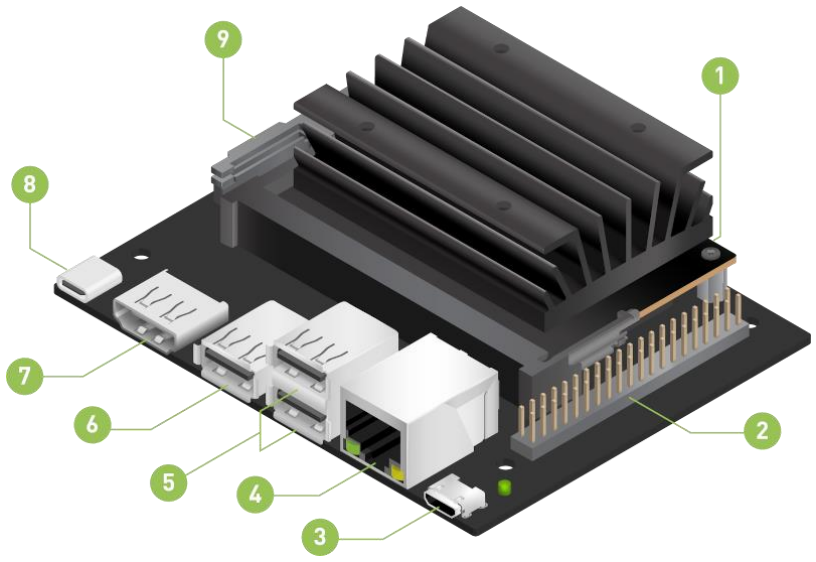

Figure 4: NVIDIA Jetson Nano representation with devices and interfaces

The micro-SD card is flashed using the SDK manager tool and the Jetson nano is set. The interaction with the development kit is through a display, keyboard and mouse or using a mode

from another computer. After the developer kit is booted and ready, the data directory will have the image data used for classification and object detection.

In Github there are pre-built docker images for the project or use with different versions of JetPack. The images are available and hosted on DockerHub, or the project can be built from scratch. The containers use Pytorch-based base containers to support training models. The transfer learning is already included in the container. The container is launched using the docker/run.sh script to use the correct container based on the version of JetPack. The data directories are mounted to include the use of cameras/display/etc in the container. After launch the container runs to execute programs and applications. The scripts will also run on x86 systems that have NVIDIA GPU however, the NVIDIA drivers must be installed to support the GPU in docker.

3.2 Image classification

With Jetson Nano DNNs are provided with image recognition and classification to identify scenes and objects. The imageNet object will accept image input and the probability of each class from the dataset. The images in the dataset were trained, and the GoogleNet and ResNet-18 models were downloaded and used in the testing stage. For instance, images of white sharks, Figure 5 are used for object detection and classification.

Figure 5: Image of great white shark used in testing

The sample image in the Figure 5 is loaded along with additional images, as in Figure 6

Figure 6: Image (hammer head shark) used for testing

For the sample images above the ResNet-18 pre-trained image classification model is available for use in ImageNet Program on Jetson Nano. Here the classification model to load can be set using the --network flag on the command line in ResNet-8. The script for loading the white shark in the ResNet-18 model is,

./imagenet.py --network = resnet-18 images/whiteshark.jpg

images/test/output_whiteshark.jpg

./ imagenet.py --network = resnet-18 images/hammerhead.jpg images/test/

output_hammerheadshark.jpg

For identifying the bird in Figure 7,

/imagenet.py --network = resnet-18 images/Americanrobin.jpg images/test/

output_Americanrobin.jpg

Figure 7: Testing for image classification, the bird in the picture

From the Figures 5, 6 and 7 it is noted the accuracy of classification is over 90%.

3.3 Object Detection

In addition, the object detection was tested using the models from the dataset. Object detection will find the objects in the fame through the extraction of their bounding-boxes. Object-detection networks can detect different objects in a frame. To perform object detection, the detectNet object is used to input the image, the outputs will consist of coordinates related to detect bounding-boxes along with their class and confidence values. detectNet is available in Python language. As earlier, pre-trained models for detection are available for download. The

default model used is the SSD-Mobilenet-v2 model trained using the MS COCO dataset to achieve real time inferencing performance with Jetson Nano. The detectNet program has the ability to locate objects in both static images and video streams.

The following command line options are used,

--network flag is used to change the detection model, default is SSD-Mobilenet-v2.

--overlay flag is an option and helps in comma separated combination involving box, labels, lines, and none.

--alpha option is used to set the alpha-blending value during overlays, with a default value 120

--threshold option value will set the value for minimum threshold to detect the image, default is set at 0.5

The output images are saved on the images/test directory if the Docker container is used. The images are easily viewable from this directory through a host device in jetson-inference/data/images.

Figure 8 provides an example to detect pedestrians using the ResNet-18 model.

Figure 8: Test for object detection model, detected pedestrian

The next Figure 9 detects a group of pedestrians irrespective of objects in the image.

Figure 9: object detection, detecting more than one pedestrian

The above example images for detecting pedestrians were done using the ResNet-18 model, the image data was obtained from the default SSD-Mobilenet-v2 model. The script used is,

./detectnet.py –network=ssd-mobilenet-v2 images/peds_0.jpg images/test/output.jpg

To detect one pedestrian as in Figure 8 the script is

./detectnet.py images/peds_1.jpg images/test/output.jpg

Likewise, the Coco 128 dataset which has images of different objects such as animals, vehicles, humans, etc., was used to detect objects from a selected image.

In case there are multiple images where one-time processing is required, the detectNet program is launched with directory path of images data using a wild card sequence as in the script,

./detectnet.py “images/peds_*.jpg” images/test/peds_output_%1.jpg

Object detection is also possible using video files and camera streams in real-time using the above techniques.

3.4 Optimizations

The image classification and object detection were further optimized using the Resnet-18, Yolo v5 and Yolo v7.

3.4.1 Resnet-18

This architecture has 18 convolution layers. The Resnet-18 model is used to train and test the model. The steps followed in training the model include,

- Load the data (data loader)
- Transform data
- Build the model, here the epochs are set, learning rate, momentum and output classes are set for the model.
- The model is trained and the best trained model is saved.
- The steps followed in testing using Resnet-18 include,
- Data loader
- Transform the data
- The saved model is loaded as input

- Test images are predicted (image classification, object detection)
- Alert if the intended object or image is found

Importantly to build the Resnet model some parameters have to be set for each use case. Since output classes in the Resnet model are high, the very last-layer of the ResNet model is changed according to the given use-case for prediction. The other parameters to build the model include epochs, learning rate momentum and the optimizer. For instance, a model having learning rate 0.001 and momentum of 0.09 will iterate over 10 epochs for binary classification and 50 epochs for multiclass classification while using the stochastic gradient descent (SDG) optimizer. The model after it is built is trained with the prescribed parameters. The model performance after each completed iteration is evaluated and model weight values saved to verify the best accuracy. In this manner, evaluations were repeated till the very last epoch and lastly, two best models were achieved, one for binary-classification and the other for multi-class classification. These entire iteration steps were performed on the Jetson Nano.

3.4.2 YOLO v5

Yolo v5 has the ability to improve optimizer, learning rate and momentum turning. Yolo 5 model optimizer algorithm with advanced target detection capability. Yolo v5 has two content security policy (CSP) structures namely CSP1_x and CSP2_x. These structures support extracting generic features from the image. Yolo v5 will self-adapt, which implies this will adapt to change the network width and by changing parameters of its own volume scale. These features in Yolo v5 guarantee good training results and high accuracy in object detection.

3.4.3 YOLO v7

This is an improved version of the DL algorithm considered in the research to ensure the feasibility of object segmentation. To deal with object segmentation in complex backgrounds this algorithm's accuracy can be further improved by adding an attention mechanism. Also, by

introducing weighted loss function semantic segmentation in images having complex backgrounds can be improved for detection accuracy. Further, YOLO v7 is faster compared to other algorithms and due to its detection speed accurate semantic segmentation is achieved quickly. For instance, pedestrians can be quickly detected in Figure 9.

Yolo V7 has four layers namely feature extraction, feature enhancement, detection and post-processing layers. In Yolo v7, the initial three layers are processed using a traditional convolution layer and the rest of the layers processed using one convolution and pooling layers. Basically, the algorithm will extract effective features from images. A CNN is used in the Yolo v7 algorithm and has three modules for convolution namely the LSTM, Dropout and Softmax output modules. Here, the LSTM will detect using the connection and full activation, where the dropout module will be added after the first two networks to avoid over-fitting. The sotmax output module is processed by the convolution to obtain the desired result. The speed and accuracy of Yolo v7 are improved when processing is done in GPU. Yolo v7 provides 120% faster processing compared to Yolo v5 with improved accuracy and speed.

3.5 Research Design

The research design follows the qualitative research approach. Qualitative research is considered for the project as it aims to provide answers on how the performance of image classification can be improved using different CNN models using different accelerator boards. This research design also helps to develop predictions, and new ideas and gather new information related to the project (Kyngäs, 2020). In addition to qualitative design, the design also considers exploratory and diagnostic approaches. An exploratory research approach is used to explore and improve the ideas surrounding image classification performance. Also, missing data related to the research project can be identified with exploratory research. The diagnostic approach to the project will evaluate the reasons for a certain phenomenon and

consists of three stages namely problem inception, diagnosis and solution (Mishra and Alok, 2022). Given the above aspects, the project follows a mixed research approach.

3.5.1 Data Analysis

The analysis of data involves the comparison study of different accelerator boards. The accelerator boards. There are various accelerator boards used to deploy DL applications that are resource-intensive. Commonly the accelerator board often make use of GPU or FPGA. CNN model will convolute filters with an input image to develop feature maps (Mittal, 2020). The convolution will involve four steps,

1. Over-laying the filter at an image location
2. Determining the product using the filter value with the corresponding value of pixel on the image
3. Adding the above product, the sum is the value obtained will be the target value of the image output
4. The above three ep are repeated for all the positions in the image

To perform the above operations the CPU is inadequate and hence accelerator using FPGA is used.

GPUs: Graphics processing units are another type of accelerator to implement the CNN model for achieving high performance. GPUs have the ability to process large data in parallel to support the algorithm to perform in terms of magnitude and faster than traditional CPUs.

Google Coral: This accelerator works using a USB port and provides a tensor processing unit (TPU) to the system. This setup enables high-speed ML inferencing on a range of systems and works easily as the connectivity is through a USB port. The platform interfaces easily with neural networks for accelerating and with embedded devices. The accelerator is enabled by an Edge TPU co-processor that provides high performance in neural network inferencing.

Nvidia Jetson Nano: Jetson Nano is an AI computer for use with embedded devices to harness AI capabilities. This is used along with embedded IoT devices to deliver the power of modern AI systems. The system comes with Jetpack SDK with libraries for accelerator, DL, graphics, multimedia, computer vision applications and so on. The NVIDIA Jetson Nano module is used in computer vision applications to perform AI-based vision tasks such as object detection, image segmentation, image classification, and so on.

Intel NCS accelerator board: Intel NCS (Neural Compute Stick) is a USB-based plug-and-play AI device or DL inference application. This device involves Intel MovidiusMyraid X VPU with 16 programmable shave cores and a dedicated neural network compute engine for hardware acceleration in DL inferences.

3.5.2 Epochs

Epoch refers to all the training set being used once to reach the total number of iterations of all the training dataset in one cycle. All the training data is exactly used once. This refers to one cycle for training the training model. The epoch is understood as the number of passes the training set is given as input to the algorithm. Here, the forward pass and backward pass is the count of one pass around the algorithm. Epoch comprises of one or more batches used to train the neural network (Oyama, Koyama and Kawasaki, 2023)

3.5.3 Batch Size

Batch size is an important hyper-parameter in ML. The hyper-parameter defines the number of parameters to work with before the internal model parameters are updated. Batch size is highly essential to ensue he model will provide peak performance. In CNN models, the batch size indicates the number of samples processed before updating the model. The epochs number is the number of complete passes through the training dataset. Here, the size of a batch must be almost equal to the number of samples in the training set (Citovsky, 2021).

3.5.4 Learning Rate

Learning rate indicated by α, a hyper-parameter used to manage the rate at which an algorithm will update or learn the values of parameter estimates. The learning rate will regulate weights of neural network related to loss gradient. In CNN models the learning rate can be configured hyper-parameter that is used in training the network. Here, the learning rate has a small positive value in range of 0.0 to 1.0. This hyper-parameter is used to control the model to quickly adapt with the problem being solved (Wienan et al., 2020).

3.5.5 CNN Models

Mask R-CNN: This model name stands for Region-based CNN (R-CNN), an extension of the Faster R-CNN algorithm for object-detection. The Mask R-CNN is an ML model also used, for instance, in segmentation tasks in computer vision. At a conceptual level, Mask R-CNN provides outputs for each candidate object, two outputs namely one class label and bounding box offset. A third branch in the network outputs a distinct object-mask. This implies the Mask R-CNN will add a branch to predict segmentation masks on each Region of Interest (RoI). RoI will be generated in parallel with the branch for classification and bounding box regression (Zhang et al., 2020b), illustrated in Figure 10.

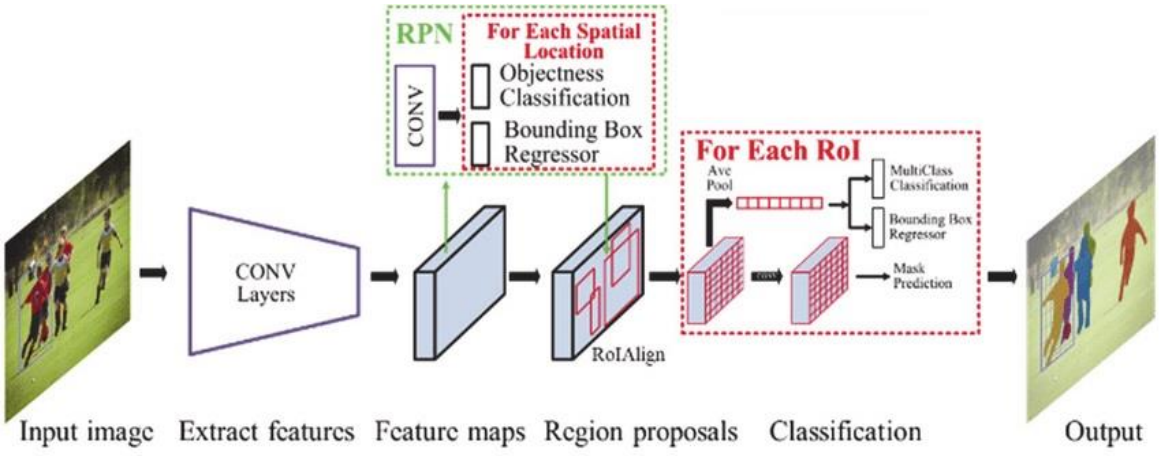

Figure 10: Image segmentation using Mask R-CNN architecture (Zhou et al., 2021)

This is an extension of the stage Faster R-CNN that has a Region Proposal Network (RPN) to recommend bounding-box for the candidate object as the first stage. The second stage will extract features from the RoI (Region of Interest) Pool or RoIPool to classify the image. The second stage is in parallel to the prediction and box offset, the Mask R-CNN will provide the binary mask for each RoI as output.

The model is based on the DL mechanism and hence the capability to classify objects as different classes, surround them with bounding boxes and create a mask for detected objects. The loss function (Podder, Bhattacharjee and Roy, 2021) is provided by,

$$L = L_{cls} + L_{box} + L_{mask}$$

where L_{cls} is the loss of classification, L_{box} is the loss of bounding-box and L_{mask} is the mask loss.

The mask branch will have one Km^2 -dimensional output for each RoI. The output will encode K binary masks of resolution $m \times m$, for each K class. Next, the per-pixel sigmoid is applied and L_{mask} is defined for the average binary cross-entropy loss. The average binary cross-entropy loss (Podder, Bhattacharjee and Roy, 2021) is given by the expression,

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \le i, 1 \le j} [y_{ij} \log \bar{y}_{ij}^{k} + (1 - y_{ij}) \log(1 - \bar{y}_{ij}^{k})]$$

Here, the label value of a cell (i, j) of any region with dimensions $m \times m$ is given by the predicted value of the k^{th} class of that cell given by $y_{ij} \cdot \bar{y}_{ij}^{k}$. In Mask R-CNN the loss function is minimised almost as zero while training the dataset, this also implies the model will perform without the problem of over-fitting.

3.5.6 Hardware and Specifications

The development boards used for comparison are:

- Google Coral
- Nvidia Jetson Nano
- Intel NCS board
- A laptop computer and Raspberry Pi 4 are used for the implementation.
- CNN models used: Mask R-CNN and YOLO
- Programming: Python or C/C++ language

The training dataset model and the inference datasets shall be implemented on each board. The performance metrics will be obtained for each board. To increase performance optimisations will be done using characteristics such as speed, accuracy of classification and energy consumption.

Increased performance shall be achieved by training the hyper-parameters and values or metrics obtained for comparison purposes. The coco 128 datasets are used in real-time object detection, the dataset used as input for training and testing to prototype different CNN models in different accelerator development boards. The CNN models used are benchmarked against each other, and after establishing benchmarking the hyperparameters are optimised. Subsequently, the hyper-parameters will be increased to improve performance on each of the accelerator boards. The accelerator boards are optimised and based on the output performance of each board the best-performing board and hyperparameters are noted. The most optimal result will indicate the board that is providing the best performance.

Chapter 4: Results and Discussions

Object detection and segmentation is done using the YOLO algorithm. YOLO will look at the image once to predict the objects present along with their position in the image. YOLO is a single CNN to predict multiple bounding boxes and class probabilities for these boxes simultaneously. As the entire image is seen, during training and testing the entire image is implicitly encoded with contextual information on classes along with the appearance. Another advantage of YOLO is that it can learn generalizable representations of objects and hence this model can be applied to new domains and can handle unexpected inputs (Diwan, Anirudh and Tembhurne, 2022). An illustration of YOLO in object detection is provided in Figure 11.

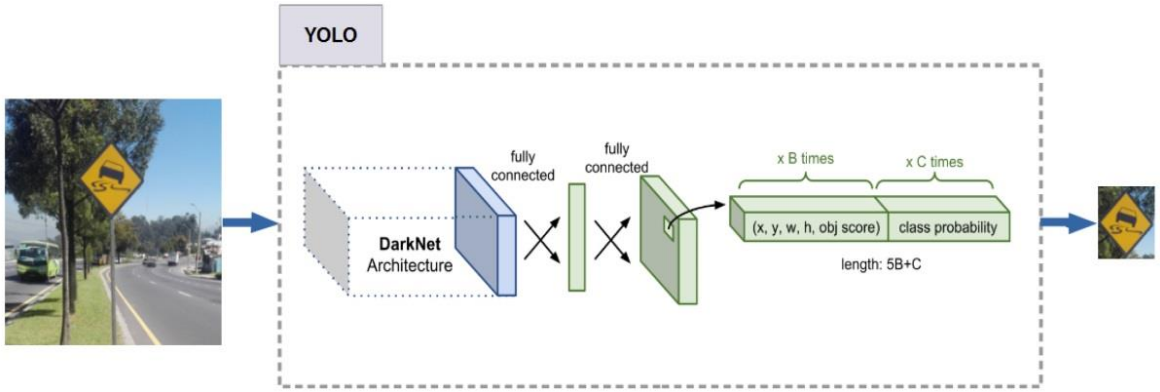

Figure 11: Object detection using YOLO. For example, the detection of road signals (Flores-Calero et al., 2024)

The detection mechanism has unified components separated into one single neural-network. The network uses features from the full-image to predict each bounding-box. YOLO can predict all the bounding boxes across different classes in the image simultaneously. This implies all the objects in the image can be classified. YOLO design supports end-to-end training and real-time speeds along with high average precision maintained (Liang Tianjiao and Hong, 2020).

For instance, an input image is first divided as a $S \times S$ grid, so the centre of the object falls in the grid the grid cell has the responsibility to detect object. The bounding box B is predicted by each grid cell along with confidence scores for these boxes. The confidence score indicates the level of confidence of the model and the accuracy decided by the model for the box predicted. Confidence is defined as,

$$\Pr(Object) * IOU_{pred}^{truth}$$

This implies that confidence is the product of the predicted object and the intersection over union (IOU) of prediction. In the absence of no object on the cell, zero will be the confidence-score. In case the object is present in the cell, the confidence-score will be equal to IOU present in the box predicted and the ground-truth. Each bounding-box will have 5-predictions represented as *x, y, w, h* and *confidence*. The coordinates (*x, y*) indicates the centre of the box in relation to boundary of the grid-cell. The width and height values are predicted based on the full image. Lastly, the confidence prediction will represent the IOU between the predicted and ground-truth boxes.

Further, every grid cell will predict *C*-conditional class probabilities. The probabilities are associated with the grid-cell having the object. Predictions can be made for one set only with the probabilities for each grid-cell irrespective of the number of boxes *B*. During the testing model, the conditional class probabilities and the individual box confidence predictions are multiplied by the expression,

$$\Pr\left(Class_i|Object\right) * Pr\left(Object\right) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth}$$

The above expression will provide class class-specific confidence score for each box. These scores will encode the probability of the class appearing in the box and on how well the predicted box fits the object.

As mentioned above, YOLO can predict several bounding boxes for each grid cell. During training, one predictor can be assigned as responsible for predicting the object in the image. For example, the road signal. This prediction will have the largest current IOU with ground truth to result in specialisation between bounding-box predictors. These aspects support each predictor to predict more accurately based on size, aspect ratio, and class of object thus improving the overall recall (Joseph et al., 2021).

Using the optimizer techniques the Jetson was trained and results discussed. Results indicate that at epoch 30 the Resnet-18 model achieved 80% accuracy for both image classification and object detection tasks. Further at epoch 65, the weights converged on 82.5% accuracy. Hence, it was noted that while training time is increased, the accuracy can be improved further while the dataset is also increased in terms of size. The Cat/Dog object from the image dataset was trained using the ResNet-18 model, shown in graph, Figure 12.

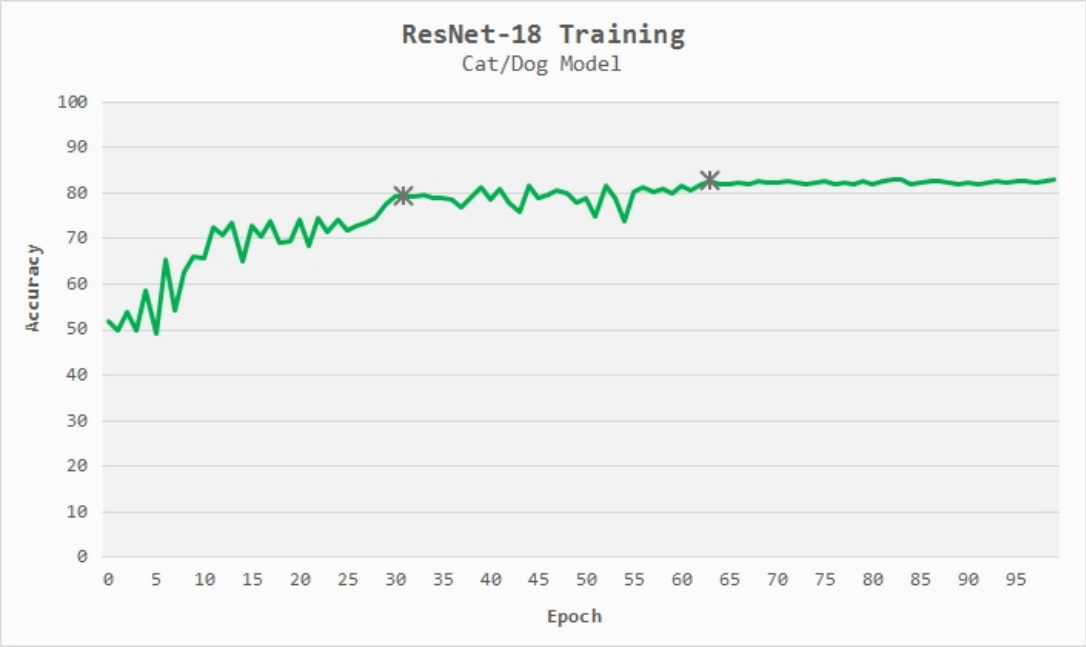

Figure 12: Training results using Resnet-18

While training it was noted that Jetson Nano was not efficient as 1 epoch took almost 15 minutes to train and provided 50% accuracy. However, after 30 epochs 80% accuracy was

achieved. However, the time consumed for training is too high. From the graph it is noted that when the number of epochs is increased the accuracy is also gradually increased. Hence more complex datasets are needed. More number of iterations must be performed to improved accuracy. In order to shorten the time consumed during training the optimizer discussed earlier were used for image classification and object detection while using Jetson Nano. The outputs for object detection and image classification outputs using test dataset were effective as it could provide over 90% accuracy as mentioned earlier using the model. The role of optimizers was important to improve the efficiency, but there is still need or more computing resources in the presented model.

Chapter 5: Conclusion

The report introduces a new technique in image classification and object detection. The use of CNN, AI, DL and ML algorithms and techniques are significant in the area of image classification and object detection. From literature it was noted that a lot of studies present the use of DL algorithms and CNN models for image segmentation. However, not much data was identified in the area of improving performance using accelerator boards and optimization. This research aims to mainly address this gap.

The report explains different CNN functions and methods and highlights the evaluation metrics used in performance measurement. The CNN models and frameworks explored are explained briefly to highlight the role of CNN in image classification problem. The methodology sections highlight the methods followed in this research along with data sources, the variables identified and data analysis techniques followed in this project. The hardware specifications of the boards are briefly described along with the steps or techniques for optimization. The optimization techniques used in the project are ResNet-18 and YOLO SDG and Adam algorithm. The image classification accuracy for some random images from the coco 128 dataset obtained were above 80% after training the model using 50 epochs. It is note that the model can improve accuracy and performance when more datasets are trained.

Future work: Further work on this topic will involve training more image datasets for classification and object detection from real world video streams. In this manner, the effectiveness of the techniques and model presented in this project will be further verified and methods validated.

References

Ahmed, N.A. (2023). *What is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained*. https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning.

Akhtar, N. and Ragavendran, U. (2020). Interpretation of intelligence in CNN-pooling processes: a methodological survey. *Neural Computing and Applications*, 32. doi:https://doi.org/10.1007/s00521-019-04296-5.

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1). doi:https://doi.org/10.1186/s40537-021-00444-8.

Bakhshi, A., Chalup, S. and Noman, N. (2020). Fast Evolution of CNN Architecture for Image Classification. In: *Deep Neural Evolution*. Singapore: Springer Science+Business Media. Springer, pp.209–229. doi:https://doi.org/10.1007/978-981-15-3685-4_8.

Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S. and Miao, Y. (2021). Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing*, 13(22), p.4712. doi:https://doi.org/10.3390/rs13224712.

Chowdhury, Y.S., Dasgupta, R. and Nanda, S. (2021). Analysis of Various Optimizer on CNN model in the Application of Pneumonia Detection. In: *3rd International Conference on Signal Processing and Communication (ICPSC)*. [online] Coimbatore, India: IEEE Explore, pp.417–421. doi:https://doi.org/10.1109/ICSPC51351.2021.9451768.

Citovsky, G., DeSalvo, G., Gentile, C., Karydas, L., Rajagopalan, A., Rostamizadeh, A. and Kumar, S. (2021). Batch Active Learning at Scale. In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*. [online] pp.11933–11944. Available at: https://proceedings.neurips.cc/paper_files/paper/2021/hash/64254db8396e404d9223914a0bd355d2-Abstract.html [Accessed 17 Apr. 2024].

Demir, F. (2022). *Deep autoencoder-based automated brain tumor detection from MRI data*. *Artificial Intelligence-Based Brain-Computer Interface*, Academic Press, pp.317–351. doi:https://doi.org/10.1016/b978-0-323-91197-9.00013-8.

Dhillon, A. and Verma, G.K. (2020). Convolutional neural network: a review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, 9(2). doi:https://doi.org/10.1007/s13748-019-00203-0.

Dhruv, P. and Naskar, S. (2020). Image Classification Using Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN): A Review. In: *Machine Learning and Information Processing. Advances in Intelligent Systems and Computing*. Springer, pp.367–381. doi:https://doi.org/10.1007/978-981-15-1884-3_34.

Diwan, T., Anirudh, G. and Tembhurne, J.V. (2022). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, 82, pp.9243–9275. doi:https://doi.org/10.1007/s11042-022-13644-y.

Du, L., Zhang, R. and Wang, X. (2020). Overview of two-stage object detection algorithms. *Journal of Physics: Conference Series*, 1544. doi:https://doi.org/10.1088/1742-6596/1544/1/012033.

Ferrer, L. (2022). Analysis and comparison of classification metrics. *arXiv preprint arXiv:2209.05355*.

Flores-Calero, M., Astudillo, C.A., Guevara, D., Maza, J., Lita, B.S., Defaz, B., Ante, J.S., Zabala-Blanco, D. and Armingol Moreno, J.M. (2024). Traffic Sign Detection and Recognition Using YOLO Object Detection Algorithm: A Systematic Review. *Mathematics*, 12(2), p.297. doi:https://doi.org/10.3390/math12020297.

Gulakala, R., Markert, B. and Stoffel, M. (2023). Rapid diagnosis of Covid-19 infections by a progressively growing GAN and CNN optimisation. *Computer Methods and Programs in Biomedicine*, 229, p.107262. doi:https://doi.org/10.1016/j.cmpb.2022.107262.

Hafiz, M.T.K., Masood, T., Jaffar, A., Akram, S. and Sohail Masood Bhatti (2023). Performance analysis of state-of-the-art CNN architectures for brain tumour detection. *International Journal of Imaging Systems and Technology*, 34(1). doi:https://doi.org/10.1002/ima.22949.

Hossain , M.R. and Timmer, D. (2021). Machine learning model optimization with hyper parameter tuning approach. *Global Journal of Computer Science and Technology*, 21(D2), pp.7–13.

Joseph, E.C., Bamisile, O., Ugochi, N., Zhen, Q., Ilakoze, N. and Ijeoma, C. (2021). Systematic Advancement of Yolo Object Detector For Real-Time Detection of Objects. In: *18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. Chengdu, China: IEEE, pp.279–284. doi:https://doi.org/10.1109/ICCWAMTIP53232.2021.9674163.

Kim, S., Lee, J., Kang, S., Lee, J. and Yoo, H.-J. (2020). A Power-Efficient CNN Accelerator With Similar Feature Skipping for Face Recognition in Mobile Devices. *IEEE Transactions on Circuits and Systems I-regular Papers*, 67(4), pp.1181–1193. doi:https://doi.org/10.1109/tcsi.2020.2966243.

Kyngäs, H. (2020). Qualitative Research and Content Analysis. In: *The Application of Content Analysis in Nursing Science Research*. Springer, pp.3–11.

Li, T., Yan, Y. and Du, W. (2022). Sign Language Recognition Based on Computer Vision. In: *2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. China: IEEE Xplore, pp.927–931. doi:https://doi.org/10.1109/ICAICA54878.2022.9844497.

Liang Tianjiao and Hong, B. (2020). A optimized YOLO method for object detection. In: *16th International Conference on Computational Intelligence and Security (CIS)*. Guangxi, China: IEEE. doi:https://doi.org/10.1109/cis52066.2020.00015.

Mandeep (2024). *What do you mean by ILSVRC dataset in Deep Learning and Why it is used ?* [online] Medium. Available at: https://medium.com/@mandeep_61901/what-do-you-mean-by-ilsvrc-dataset-in-deep-learning-and-why-it-is-used-1be8b692994b#:~:text=The%20ImageNet%20Large%20Scale%20Visual [Accessed 17 Apr. 2024].

Marcu, D.C. and Grava, C. (2021). The impact of activation functions on training and performance of a deep neural network. In: *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)*. Romania: IEEE. doi:https://doi.org/10.1109/emes52337.2021.9484108.

Meel, V. (2021). *What is the COCO Dataset? What you need to know in 2021*. [online] viso.ai. Available at: https://viso.ai/computer-vision/coco-dataset/ [Accessed 16 Apr. 2024].

Mishra , S.B. and Alok, S. (2022). *Handbook of research methodology* . Educreation publishing.

Mishra, M. (2020). *Convolutional Neural Networks, Explained*. [online] Medium. Available at: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 [Accessed 28 Mar. 2024].

Mittal, S. (2020). A survey of FPGA-based accelerators for convolutional neural networks. *Neural Computing and Applications*, 32(4), pp.1109–1139. doi:https://doi.org/10.1007/s00521-018-3761-1.

Oh, S., Choi, J. and Kim, J. (2020). A Tutorial on Quantum Convolutional Neural Networks (QCNN). In: *IEEE Xplore*. Jeju, South Korea: IEEE, pp.236–239. doi:https://doi.org/10.1109/ICTC49870.2020.9289439.

Oyama , N., Koyama , S. and Kawasaki, T. (2023). What do deep neural networks find in disordered structures of glasses? *Frontiers in Physics* , 10. doi:https://doi.org/10.3389/fphy.2022.1007861.

Peng, X., Zhang, X., Li, Y. and Liu, B. (2020). Research on image feature extraction and retrieval algorithms based on convolutional neural network. *Journal of Visual Communication and Image Representation*, 69, p.102705. doi:https://doi.org/10.1016/j.jvcir.2019.102705.

Podder, S., Bhattacharjee, S. and Roy, A. (2021). An efficient method of detection of COVID-19 using Mask R-CNN on chest X-Ray images. *AIMS Biophysics*, 8(3), pp.281–290. doi:https://doi.org/10.3934/biophy.2021022.

Qin, J., Pan, W., Xiang, X., Tan, Y. and Hou, G. (2020). A biological image classification method based on improved CNN. *Ecological Informatics*, 58. doi:https://doi.org/10.1016/j.ecoinf.2020.101093.

Sharma, S., Mehra, R. and Kumar, S. (2020). Optimised CNN in conjunction with efficient pooling strategy for the multi-classification of breast cancer. *IET Image Processing*, 15(4). doi:https://doi.org/10.1049/ipr2.12074.

Shyam, R. (2021). Convolutional Neural Network and its Architectures . *Journal of Computer Technology & Applications* , 12(2). doi:https://doi.org/10.37591/JoCTA.

Simic, M. (2024). *Intersection Over Union for Object Detection*. [online] Baeldung. Available at: https://www.baeldung.com/cs/object-detection-intersection-vs-union [Accessed 28 Mar. 2024].

Singh, S., Yadav, A., Jain, J., Shi, H., Johnson, J. and Desai, K. (2024). Benchmarking Object Detectors with COCO: A New Path Forward. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2403.18819.

Srivastava, H. and Sarawadekar, K. (2020). A Depthwise Separable Convolution Architecture for CNN Accelerator. In: *IEEE Applied Signal Processing Conference (ASPCON)*. Kolkata, India: IEEE. doi:https://doi.org/10.1109/aspcon49795.2020.9276672.

Stankovic, L. and Mandic, D. (2023). Convolutional Neural Networks Demystified: A Matched Filtering Perspective-Based Tutorial. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(6), pp.1–15. doi:https://doi.org/10.1109/tsmc.2022.3228597.

Sultana, F., Sufian, A. and Dutta, P. (2020). A Review of Object Detection Models Based on Convolutional Neural Network. In: *Advances in Intelligent Systems and Computing*. Singapore: Springer, pp.1–16. doi:https://doi.org/10.1007/978-981-15-4288-6_1.

Sun, Y., Xue, B., Zhang, M., Yen, G.G. and Lv, J. (2020). Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Transactions on Cybernetics*, 50(9), pp.3840–3854. doi:https://doi.org/10.1109/TCYB.2020.2983860.

Taye, M.M. (2023). Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. *Computation*, 11(3), p.52. doi:https://doi.org/10.3390/computation11030052.

Valarmathi, G., Suganthi, S.U., Subashini, V., Janaki, R., Sivasankari, R. and Dhanasekar, S. (2021). CNN algorithm for plant classification in deep learning. *Materials Today: Proceedings*, 46(9), pp.3684–3689. doi:https://doi.org/10.1016/j.matpr.2021.01.847.

Vujovic, Ž.Đ. (2021). Classification Model Evaluation Metrics. *International Journal of Advanced Computer Science and Applications*, 12(6). doi:https://doi.org/10.14569/ijacsa.2021.0120670.

Wang, P., Fan, E. and Wang, P. (2020). Comparative Analysis of Image Classification Algorithms Based on Traditional Machine Learning and Deep Learning. *Pattern Recognition Letters*, 141, pp.61–67. doi:https://doi.org/10.1016/j.patrec.2020.07.042.

Weinan, E., Ma, C., Wojtowytsch, S. and Wu, L. (2020). Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't. *arXiv:2009.10713v3. Cornell University*. doi:https://doi.org/10.48550/arxiv.2009.10713.

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A.K. and Almotairi, S. (2022). A Comparison of Pooling Methods for Convolutional Neural Networks. *Applied Sciences*, 12(17), p.8643. doi:https://doi.org/10.3390/app12178643.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D. and Saeed, J. (2020). A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction. *Journal of Applied Science and Technology Trends*, 1(2), pp.56–70. doi:https://doi.org/10.38094/jastt1224.

Zhang, S., Cao, J., Zhang, Q., Zhang, Q., Zhang, Y. and Wang, Y. (2020a). An FPGA-Based Reconfigurable CNN Accelerator for YOLO. In: *IEEE 3rd International Conference on Electronics Technology (ICET)*. Chengdu, China: IEEE. doi:https://doi.org/10.1109/icet49382.2020.9119500.

Zhang, Y., Chu, J., Leng, L. and Miao, J. (2020b). Mask-Refined R-CNN: A Network for Refining Object Details in Instance Segmentation. *Sensors*, 20(4), p.1010. doi:https://doi.org/10.3390/s20041010.

Zhou, Y.-C., Hu, Z.-Z., Yan, K.-X. and Lin, J.-R. (2021). Deep Learning-Based Instance Segmentation for Indoor Fire Load Recognition. *IEEE Access*, 9, pp.148771–148782. doi:https://doi.org/10.1109/access.2021.3124831.