# CluFy: low power object classification model based on object estimation

## Abstract

*Due to the dramatic increase in the volume of image data production and the transformation of a large amount of computation to the edge, the design of lightweight and low-power image processing models that can run on edge devices with limited resources is very important. In this article, we describe an algorithm to reduce the size of image processing models. This algorithm uses the initial estimation technique that is carried out in the human brain and reduces the search space for object recognition in such a way that it reduces the space of classification classes with a simple processing model, then with a more accurate model ,which trained on small number of classes, try to classify it. In shortly means that instead of using a large model that recognizes a large number of classes but does not have the ability to deploy on edge devices, we use a number of small models that each recognize a small number of classes, and in total, the sum of classes , which classify by these models, is equal to the larger model.*

## 1. Introduction

AI and machine learning, particularly CNN, are essential in the advancement of smart cities. These techniques are used efficiently and effectively [20] in various IoT applications, such as intelligent transportation systems, unmanned aerial vehicles[1], [12], smart grids, surveillance, and healthcare systems of a smart city[27]. IoT devices that generate large amounts of data are a vital component of these smart systems. The aforementioned techniques provide cutting-edge[41], learning and decision-making abilities by analyzing big data to support intelligent systems[27]. The rise of more advanced surveillance[43] camera technologies and their improved processing abilities also provide many benefits in the field of real-time image and video analysis, including object classification[28], [26], i.e., object tracking, and action recognition.

Object classification is a crucial task in computer vision, and real-time applications of it require high accuracy and low power consumption for use on embedded and edge-IoT devices[7], [40]. Traditional object classification methods that employ region-based and sliding window algorithms have low accuracy and high computation time. Advances in object classification algorithms using convolutional neural networks (CNNs) have led to their application in various computer vision tasks, including image classification, human behavior analysis, face recognition, and autonomous driving. However, the rapid development of CNN architectures and algorithms has also led to increased challenges in computation[15], [17], [3], [42], [4]. The large number of parameters required to implement CNN network layers increases computational complexity[17], [37], [13], [10], [36]. Additionally, CNN architectures used in object classification algorithms have different layer parameters such as kernel size and number of channels, making it difficult to design generic computing hardware for use on embedded systems. Greater hardware resources are needed to implement deeper CNN architectures for improved classification accuracy[32].

The resource consumption of object classification models depends on several factors, such as:

1. The size and complexity of the model: Larger and more complex models, such as the RCNN family of models, tend to require more resources than smaller and simpler models, such as YOLO.

2. The input image size: The larger the input image size, the more memory is required to process the image and the more computation is required to generate object classification.

3. The number of object classes: The more object classes the model is trained to detect, the more memory is required to store the model's parameters and the more computation is required to make predictions.

4. The batch size during inference: The larger the batch size during inference, the more memory is required to store intermediate computations and the more computation is required to process the batch.

In the topic of size and complexity of the CNN model, it includes several layers of convolution and pooling, followed by a fully-connected layer. These layers differ significantly in terms of the number of computations required. Many iterative calculations are necessary to process the large parameters in each layer. Convolution is a fundamental operation that makes up the majority of computations in a CNN model. As it will discuss in section3.1 the number of computations required in the fully-connected layers significantly more larger than the convolutional layers. This leads to a high demand for multiply-accumulate (MAC) operations, which results in high execution time, power consumption, and area overhead in this layer. This increased complexity makes it difficult to deploy CNN models on embedded platforms. High-throughput MAC units are used to process convolution operations, and the total execution time of the computation depends on the speed of the MAC unit[35]. These limitations have led to the development of new computing architectures that aim to reduce the execution latency of the convolution layer in CNN, power consumption, and hardware resources.

In addition The trade-off between memory usage and accuracy is a common problem in object classification, as larger and more complex models tend to be more accurate but also require more resources. One solution to this problem is to use model compression techniques, such as pruning and quantization, to reduce the model size and memory usage without sacrificing too much accuracy. Another solution is to use hardware accelerators such as GPU or TPU to speed up the computation and reduce memory usage.The size of an artificial neural network (ANN) model can have a significant impact on its resource consumption, particularly in terms of memory usage and computation time.It's worth noting that the model size doesn't only depend on the number of parameters, but also on the architecture of the model and the number of layers, the number of neurons per layer, and the dimensionality of the input.

## 2. Motivation

Today, due to the increase in the number of CCTVs in the city, a large amount of image data is produced every second in the world, which cannot be processed by humans, and only with the help of image processing models, this amount of information can be processed. In addition, due to the large volume of image data, transferring them to a central server and processing them by image processing models has many challenges. For this reason, it is very important to design light and small models of image processing that have the ability to run on edge devices that have limited resources. One of the ways to reduce the size of image processing models is to use the image processing algorithm of the human brain. The human brain uses a variety of techniques to classify objects, and one of these is the use of preliminary estimate objects. In the human brain, first by observing an object, it makes a preliminary estimate of the class of that object by using its appearance characteristics, and limits the space of object class search states from the millions of object classes learned over the years to a much smaller number of classes. And then, by using a more detailed observation, it detects the exact class of the observed object among the limited number of classes.

By using this technique and limiting the state space of classes, we can reduce our image processing model. For example, in the classification with CNN model that has been trained on the microsoft coco dataset [23] which contain of 80 classes, if the input object is a "table" and we remove other objects such as an "airplane", "engine","person" and etc. from its classification space, the size of the last layer of the CNN model will be decrease because the last layer is a fully connected layer and its size is equal to the number of recognized classes. And also the classification operation is performed between a smaller number of classes, which by itself solves our need to use complex models and we can use simpler models which is accurate like complex models .In this article, we explain this idea and its implementation.

## 3. Background And Literature Review

In this section, we give a brief explanation about some famous algorithms in computer vision that are used in classification applications.

### 3.1. Convolutional Neural Network (CNN)

A convolutional neural network (CNN) is a type of neural network architecture specifically designed to process data that has a grid-like topology, such as an image. A convolutional neural network architecture typically contains the following layers

1. Convolutional layers: These layers apply a set of filters to the input data, creating feature maps that capture different aspects of the input. The filters are typically small, and slide over the entire input, computing dot products between the entries of the filters and the input at each position.

2. Pooling layers: These layers are used to down-sample the data, reducing the dimensionality and helping to prevent overfitting. Common types of pooling include max pooling, where the maximum value in a region of the input is taken, and average pooling, where the average value in a region of the input is taken.

3. Normalization layers: These layers are used to normalize the activations of the previous layer, to make the training process more stable. Batch normalization and layer normalization are the most common types of normalization layers.

4. Activation layers: These layers apply a non-linear function to the input, such as a rectified linear unit (ReLU) or a sigmoid function. These layers introduce non-linearity in the network, allowing it to learn more complex functions.

5. Fully connected layers: These layers are used to classify the features extracted by the convolutional layers. They connect every neuron in one layer to every neuron in another layer.

6. Dropout layers: These layers are used to prevent overfitting by randomly dropping out a certain percentage of the neurons during training.

The architecture of a CNN can vary depending on the specific problem, but these layers are commonly used in many architectures. Convolutional neural network (CNN)model size is directly related to the size of the its layers specialy fully connected layer and their number. As the number of neurons in a layer increases, the number of parameters in that layer increases the model size and power usage decrease. Similarly, if we use simple model ,which consists of a small number of layers instead of using complicated model like DensNet model, for our work the model size and power usage will decrease again because of decoration in number of layer which affect on number of parameters. overlay model size directly related to number of model parameters and the total number of parameters in a CNN can be calculated by summing up the number of parameters in all the layers. The number of parameters in a layer is determined by the number of neurons in that layer, and the number of connections between neurons. In fully connected layers, each neuron is connected to every neuron in the previous layer which this layer has a large part of the size model. The size of a fully connected layer in a convolutional neural network (CNN) is determined by the number of neurons in that layer which in classification application it is related to number of classes .

## 3.2. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm that is used for classification and regression tasks. The main idea behind SVM is to find a hyperplane that maximally separates the different classes in the feature space. The points closest to the hyperplane are called support vectors and have the greatest impact on the position of the hyperplane.

In classification tasks, SVM finds a hyperplane that separates the different classes by maximizing the margin, which is the distance between the hyperplane and the closest data points from each class. In two-class classification problems, the goal is to find a hyperplane that separates the two classes with the widest possible margin.

SVM algorithm can also be used for non-linearly separable problems by using kernel trick. This technique allows the algorithm to map the input data to a higher-dimensional space, where a linear hyperplane can be used to separate the classes. Some common kernels used in SVM are linear, polynomial, and radial basis function (RBF) kernels.

SVM can also be used for regression tasks, in this case the algorithm finds a hyperplane that best fits the data points. This approach is known as Support Vector Regression (SVR).

SVM algorithm is known for its high accuracy and ability to handle high-dimensional and non-linearly separable data. However, it can be sensitive to the choice of kernel function, and the optimal hyperparameters can be difficult to find, also it can be computationally expensive for large datasets. Support Vector Machine (SVM) classifies data by finding a hyperplane that maximally separates the different classes in the feature space. The process can be broken down into the following steps:

1. Data preparation: The input data is first prepared by converting it into a suitable format for the SVM algorithm to process. This typically involves normalizing the data and transforming it into a high-dimensional feature space using a kernel function.

2. Finding the optimal hyperplane: SVM finds the hyperplane that maximally separates the different classes by maximizing the margin, which is the distance between the hyperplane and the closest data points from each class. This is done by solving a quadratic optimization problem, which involves finding the weights and bias of the hyperplane that minimize the classification error.

3. Making predictions: Once the optimal hyperplane is found, new data points can be classified by determining which side of the hyperplane they fall on. This is done by computing the dot product of the new data point with the weights of the hyperplane and adding the bias. If the result is positive, the data point is classified as belonging to one class, and if it is negative, it is classified as belonging to the other class.

4. Handling non-linearly separable problems: In case the data is not linearly separable, SVM uses the kernel trick to map the data to a higher-dimensional feature space, where a linear hyperplane can be used to separate the classes. Some common kernels used in SVM are linear, polynomial, and radial basis function (RBF) kernels.

It's worth noting that the above explanation is based on the binary classification case, the multi-class classification can be handled using one-vs-all or one-vs-one methods.

## 3.3. K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a non-parametric and lazy learning method used for classification and regression tasks.

The basic idea behind the KNN algorithm is to find the k-number of training examples that are closest to the new input data point, and then use the majority class among those k-nearest neighbors as the prediction for the new data point. The number of nearest neighbors to consider, k, is a user-defined parameter.

Here are the main steps of the KNN algorithm:

1. Data preparation: The input data is first prepared by converting it into a suitable format for the KNN algorithm to process. This typically involves normalizing the data and transforming it into a high-dimensional feature space.

2. Distance calculation: For each training example, the distance between it and the new input data point is calculated using a distance metric such as Euclidean distance.

3. Finding the k-nearest neighbors: The k-nearest neighbors are found by sorting the training examples based on their distance to the new input data point and selecting the top k closest examples.

4. Making predictions: Once the k-nearest neighbors are found, the majority class among them is used as the prediction for the new data point. In case of regression problem, the prediction would be the average of the k-nearest neighbors.

### 3.4. Object Classification

Object classification is a computer technology related to computer vision and image processing that deals with classification the objects of a certain class (such as humans, buildings, or cars) in digital images and videos. With the development of deep learning, object classification has gained lots of improvements and become a core technology in many real-world applications such as self-driving cars, surveillance systems, and object tracking.

There are several famous deep neural network (DNN) models that have been developed for object classification:

1. RCNN (Regions with CNN features): The RCNN family of models, which includes Fast R-CNN and Faster R-CNN, uses a two-stage approach for object classification. The first stage generates region proposals, and the second stage classifies the regions using CNN features.

2. YOLO (You Only Look Once): YOLO is a one-stage approach for object classification that directly predicts bounding boxes and class probabilities for the objects in an image. YOLOv3 is the latest version of YOLO which has improved accuracy and faster speed.

3. SSD (Single Shot MultiBox Detector): SSD is another one-stage approach for object classification that uses a single CNN network to predict bounding boxes and class probabilities for objects in an image.

4. RetinaNet: RetinaNet is a one-stage approach for object classification that addresses the problem of class imbalance in datasets by using a focal loss function.

5. Mask RCNN: Mask RCNN is an extension of Faster RCNN that also predicts a binary mask for each instance of an object in addition to object classification and bounding box.

6. FPN (Feature Pyramid Network): FPN is a feature extraction architecture that builds an pyramidal feature hierarchy by adding high-resolution features to lower-resolution features. It's often used as a backbone in many object classification models.

These are some of the most popular DNN models for object classification, but new ones are constantly being developed and improved upon.

## 4. Proposed Design

In this section, we want to explain our own design for reducing the number of classes from which the classification operation should be performed, because as it was explain in the previous section; Reducing the number of classes not only reduces the model size but also reduces the energy consumption.As shown in Fig. 1, our design consists of two main layers: clustering and classifying which named it CluFy model. We have used mnist and microsoft coco dataset for our work. In the following, we will explain this layers.

### 4.1. Clustring Layer

As you know, the object classification models classify the input image among the classes that have already been seen in training phase. We have used microsoft coco dataset for our work which includes **C**=80 classes. In the clustering step, we divide these **C** classes into **K** categories by using the clustering algorithm. Each category is a set of images in a certain number of classes. Here we used the Kmeans algorithm with **K** = 8 for clustering our dataset, we will explain how to get the **K** value in the next section 4.4. As it explained in section 3.4, the object classification models classify the input image among the classes that have already been seen in training phase. We have used microsoft coco dataset for our work which includes **C**=80 classes. In the clustering step, we divide these **C** classes into **K** categories by using the clustering algorithm shown in Fig 2. Each category is a set of images in a certain number of classes shown in Fig 3. Here we used the Kmeans algorithm with

**K**= 8 for clustering our dataset, we will explain how to get the **K** value in the next section.

In next step at the end of the clustering the classes of dataset, we train the model, which named it Clu model as shown in Fig 4, on these **K** clusters. After completing the training phase of Clu model, it has ability to recognize the cluster for each input image and send cluster whit maximum likelihood to next layer(classifying) . In short, the input image class is layer of the cluster that the Clu model has recognized. By doing this, the number of classes that are involved in the classification of the input image will actually decrease. In our work we use SVM algorithm for Clu model and train it on **K**=8 cluster. for example in our work on microsoft coco dataset which we cluster it in to 8 clusters you can see in Table 1 that class of "toaster" and class of "oven" are in same cluster and also in Table2 represent the number of classes in each cluster.

## 4.2. Classifying Layer

In this section, for each cluster, a separate model is trained on the classes of that cluster in such a way that it can detect its class for each image belonging to that cluster shown in Fig 5. We name this model Fy model and MobileNet[11] has been used for Fy model. For each cluster shown in Fig 6 , We have trained the Fy model on cluster images. At the end, this layer get the input image and its related cluster ,which was determined in previous layer, as inputs and fetched the related Fy model for classifying the input image.in shortly, we have some simple classification model,shown in Fig 6 ,which classify small number of classes and use one of the for related application instead of fetch classification model which classify large number of classes and has complicated model structure.

## 4.3. Error Propagation

As it is known, because our design has a pipeline structure, the error of the clustering layer affects the classification layer and reduces the overall accuracy. To prevent this, we set an $\alpha$ parameter. In this way, if the maximum likelihood of determined cluster ,which it is output of Clu model, for input image is lower than the $\alpha$ value, we give two clusters as output to the classification layer, and in the classification layer, instead of running one Fy model, two Fy models are fetched and run on the input image, one related to the first cluster and the other related to second cluster. The output class is obtained from the following Eq1. $\alpha$ value is also determined by using validation data.

$$output = \begin{cases} a & OFy1 \times OCly1 > OFy2 \times OCly2 \\ b & OFy2 \times OCly2 > OFy1 \times OCly1 \end{cases} \quad (1)$$

Where $OCly1$ is the maximum cluster likelihood and $OFy1$ is the first cluster related Fy model output's likelihood and $OCly2$ is the second maximum cluster likelihood and $OFy2$ is the second cluster related Fy model output's likelihood.$a$ is the output of first cluster related Fy model output and $b$ is the output of second cluster related Fy model output.

## 4.4. Calculate K Number

To determine the value of **K**, we must first see the effect of this number on the output accuracy and the resources utilization to implement our designed for recognizing the class of the input image. For this purpose, we have implemented CluFy model for **K** from 1 to 10 on the mnist classification application. In this example, our Clu model is SVM and Lenet-5 is Fy model.

Chart 8 is model size chart. As you can see, with the increase of **K**, the size of the model first increases and then finally decreases, because when the value of **K** =1, it means that we have just a classifying layer and there is no clustering layer, so the overall size of the design is only related to the Fy model. As the value of **K** increases, the size of the Fy model decreases because the number of classes decreases and finally with the value of **K** =10, which is equal to the total number of classes, in the clustering layer the class of the input image is determined, which means that there is no classifying layer and because the Clu model is much simpler than the Fy model, the final size of the design is small.

In the chart 7, you can see the graph of accuracy on the**K** number. As you can see, when **K** is equal to one and there is no clustering layer, the total accuracy is equal to the Fy model accuracy, and with the increase of the value of **K**, the total accuracy first increases due to the decrease in the number of classes, and finally with The value of **K** = 10, which is equal to the total number of classes, the total accuracy is equal to the accuracy of the Clu model, because this model is much simpler than the Fy model, the total accuracy is much lower than the total accuracy in **K**=1. It should also be added that if there is a cluster with one after clustering the classes of the dataset, the accuracy of recognizing that class is equal to the accuracy of the Clu model. Because this model is simpler than the Fy model it is not very accurate, so in determining the value of K in clustering process, care must be taken that no cluster contains a single class and the number of classes in each cluster is at least two. You have seen that the value of K is a trade-off parameter between model size and total accuracy, that in mnist classifying, the value of **K**=4, and by doing the same procedure in microsoft coco, the value of **K** is equal to 8.

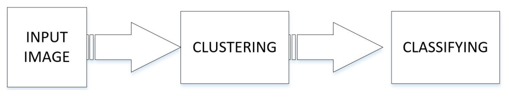

Figure 1. overall structure of our design(CluFy model).

Table 1. cluster ID of each class of microsoft coco.

| index | class | cluster ID |
|---|---|---|
| 1 | chair | ClID-3 |
| 2 | pottedplant | ClID-2 |
| 3 | diningtable | ClID-5 |
| 4 | couch | ClID-0 |
| 5 | bed | ClID-2 |
| 6 | toaster | ClID-2 |
| 7 | sink | ClID-3 |
| 8 | oven | ClID-2 |
| 9 | refrigerator | ClID-3 |
| 10 | microwave | ClID-4 |
| 11 | person | ClID-1 |
| 12 | skateboard | ClID-8 |
| 13 | surfboard | ClID-2 |
| 14 | frisbee | ClID-8 |
| 15 | snowboard | ClID-2 |
| . | . | . |
| . | . | . |
| . | . | . |
| 75 | clock | ClID-5 |
| 76 | teddybear | ClID-7 |
| 77 | hairdrier | ClID-6 |
| 78 | book | ClID-3 |
| 79 | scissors | ClID-2 |
| 80 | vase | ClID-4 |

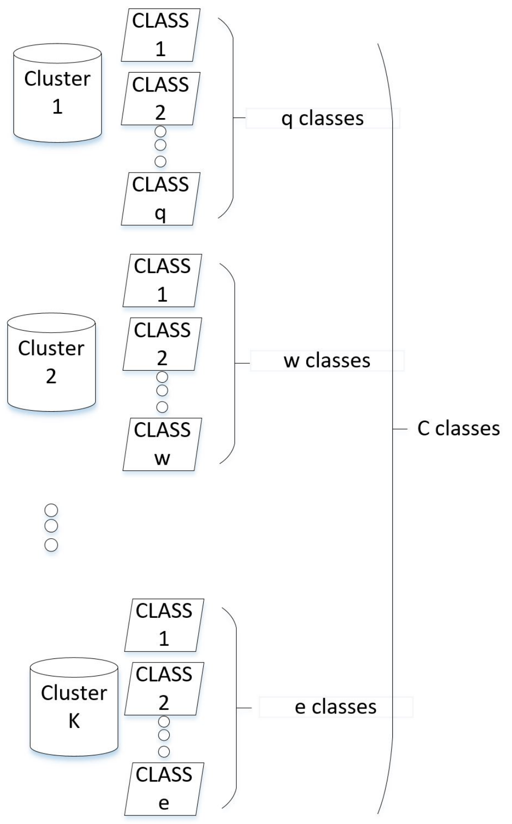

Figure 3. structure of all of clusters which each of them contain of some number of classes .

Table 2. number of classes in each cluster.

| index | cluster ID | number of class |
|---|---|---|
| 1 | ClID-0 | 9 |
| 2 | ClID-1 | 8 |
| 3 | ClID-2 | 11 |
| 4 | ClID-3 | 7 |
| 5 | ClID-4 | 12 |
| 6 | ClID-5 | 10 |
| 7 | ClID-6 | 11 |
| 8 | ClID-7 | 12 |

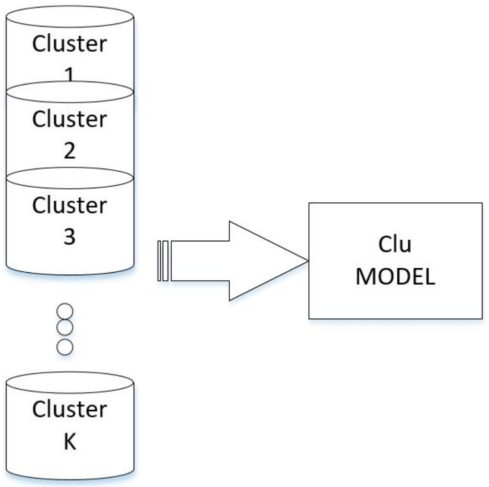

Figure 4. training Clu model for determining the related cluster of input image .

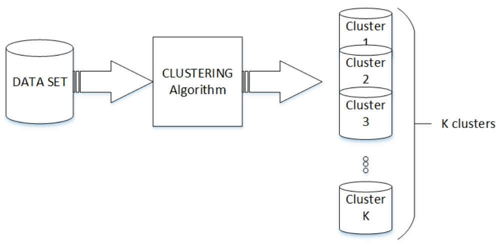

Figure 2. clustering the dataset classes.

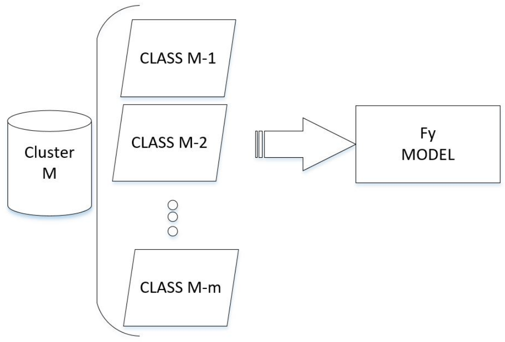

Figure 5. traing the Fy model on Mth cluster for classifying the input image(this cluster consist of m classes).

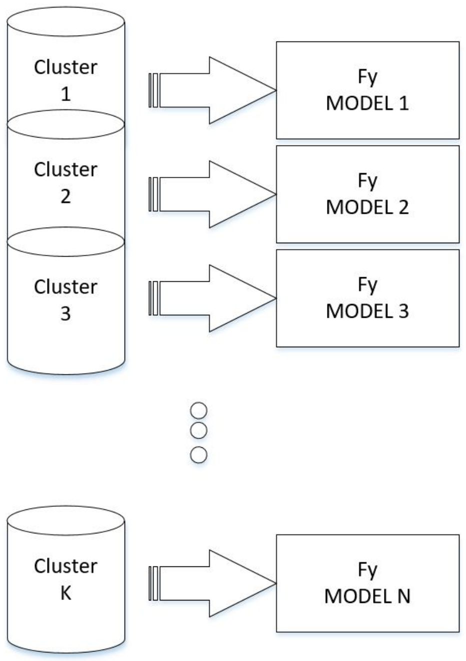

Figure 6. each cluster has its Fy model.

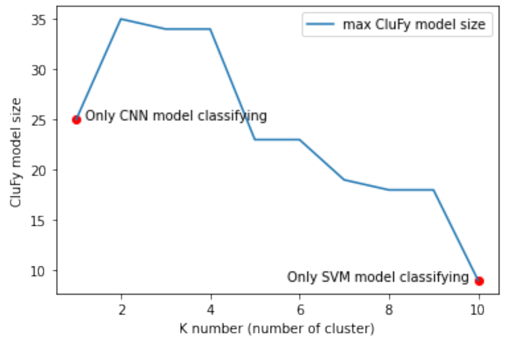

Figure 7. chart of accuracy on K number.

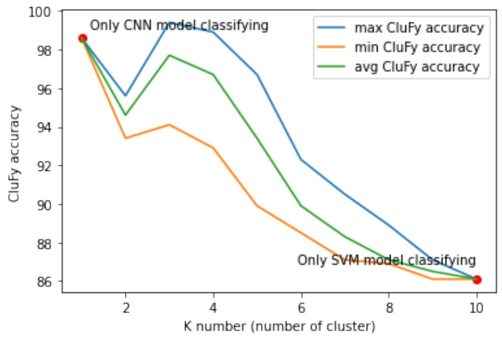

Figure 8. chart of model size on K number.

## 5. Evaluation And Comparison

As it was said in the previous part,CNNs are composed of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers are responsible for extracting features from the input data, while the pooling layers are used to down-sample the data, reducing the dimensionality and helping to prevent overfitting. The fully connected layers are used to classify the features extracted by the convolutional layers. The overall architecture is designed to automatically and adaptively learn spatial hierarchies of features from input images. the size of the CNN model decreases with the shrinking of the space of the classes. the size of fully connected layers IN CNN is equal to the size of the number of classes therefor by decreasing number of classes model size will be decrease.In the following we will compare our model in terms of accuracy, model size, and Mac operation, with Yolo and Google Net models which are very famous models for classification application with large number of class size.

First of all we explain our CluFy architecture. as it proposed in previous section CluFy model consist of two layer:clustering and classifying layers. in clustering layer we use SVM model with RBF kernel which train on microsoft coco dataset that clustering its classes by Kmeans algorithm Shown in Table1. this SVM model can determined Cluster ID for each input image.In classifying layer we train MobileNet model for each cluster till classifying input image.Each of the MobileNet model has been trained on the classes of microsoft coco dataset which have same cluster ID with the cluster ID of the MobileNet model.for example MobileNet model which trained for clssifying classes with cluster ID =ClID-0 could classify =[”couch” ,”bench”,”boat”,”handbag”,”cake”,”hotdog”, ”backpack”,”handbag”,”suitcase”].

In Table 3 we compare CluFy model size in microsoft coco classification application with YOLO-V7 and GoogleNet on same application.The CluFy model size which reported in the Table 3 is in the situation that the

Table 3. compare CluFy model size and number of MAC operation with GoogleNet and YOLOV7.

| model | model size(MB) | million MAC) |
|---|---|---|
| CLuFy | 28 | 763 |
| YOLO-V7 | 54 | 2075 |
| GoogleNet | 144 | 1550 |

Table 4. compare CluFy model accuracy with GoogleNet and YOLOV7.

| model | mAP(%) | recall(%) |
|---|---|---|
| CLuFy-Clu | 79.8 | 80.7 |
| CLuFy-Fy | 88.6 | 90.1 |
| CLuFy-Totally | 78.9 | 79.9 |
| YOLOV7 | 86.8 | 86.9 |
| GoogleNet | 82.3 | 85.1 |

$\alpha$ parameter (4.3) is less than its limit and it is necessary to fetched two Fy models.In terms of conditions, the CluFy model has a smaller size than YOLOV7[39] and GoogleNet.In number of MAC operation GoogleNet[37] has $2X$ more million MAC operation than CluFy and YOLOV7 has $3X$ more, which means that CluFy energy consumption is half of GoogleNet and third of YOLOV7.

In the following, we compare the mean average precision of the CluFy model with YOLOV7 and GoogleNet models , which we have trained them on the microsoft coco dataset with 80 classes. As you can see in the Table4, the CluFy model has lower accuracy compared to other models. Although by reducing the number of classes, Fy model in classifying layer alone is more accurate than other more complex models like GoogleNet and YOLOV7, but due to the error propagation that caused by the pipeline structure of the CluFy model, the accuracy of the entire CluFy model decreases. In other versions of the CluFy model, we design an algorithm to prevent error propagation.

## 6. Related Works

This section highlights the various architectures of accurate and approximate models. Models that are designed for low-power computation concentrate on the convolution operations and typically consist of a multiplier, an adder, and an accumulator register.[21] Cowlishaw et al. proposed a hardware implementation of decimal arithmetic units for commercial and financial purposes to address the issues with decimal to binary conversion errors[5]. However, this approach results in slower execution time compared to binary arithmetic implementations.

In embedded image processing, multiplication is a more computationally demanding operation than other arithmetic operations, and the size of the model affects the volume of multiplication computation. To address this problem,[33] Robison suggested using a fused multiply-add (FMA) operation, which reduces the complexity of the multiplication operation. The FMA processing chips are commercially available in many processors such as IMB[29], HP[19], and Intel[8]. A hardware design of a fully parallel FMA unit for decimal floating-point operations has been presented by Samy et al[34]. The proposed unit is designed specifically to execute the fused multiply-add operation, but it can also perform addition, subtraction, and multiplication operations separately. [25], [30] Another algorithm based on reversible logic has been introduced to reduce the number of logic gates, eliminate constant inputs and outputs, and simplify the hardware circuitry. This results in lower power consumption and faster execution time compared to existing architectures. However, the accumulation operation in these architectures takes longer. Sharif and[35] Prasad present a 64-bit unit that employs the Vedic Multiplier and Ripple Carry Adder to decrease the hardware area overhead. The study compared the performance of the Wallace multiplier and Vedic[31] multiplier and found the latter to be more efficient in terms of area. In addition, a low-delay, area-efficient MAC unit has been introduced for exponent addition in single-precision floating-point operations.

Recently, approximate computing has become a popular low-power design method in hardware that enhances performance, reduces power consumption, and optimizes area utilization. Several researchers [18], [9], [14], [2] have developed hardware architectures based on approximate computing to take advantage of these benefits. Kim et al [16]. conducted a comprehensive examination of different approximate architectures used for CNN operations and found that approximation of multiplication while keeping addition exact is crucial in reducing the overall approximation error. Kulkarni et al[18] . proposed a multiplier that calculates the final result by combining the outputs of approximate partial product units. The main drawback of this work is its reduced performance when dealing with larger input sizes. To address this issue, Hashemi et al[9]. introduced a multiplier unit called ”drum” that chooses only the most important bits of the operands to process the multiplication and discards the rest to simplify the hardware. This approximation reduces the size of the multiplier to half the length of the operands, but also increases latency and power consumption. Imani et al[14]. proposed an approximate floating-point multiplier that saves the expensive multiplication of the fractional part by discarding one of the mantissae[2]. This design balances area and power consumption but increases the error rate by up to 50%. Separate hardware has been created to allow for accuracy tuning and recognize high output errors. Boroumand et al[22], [24], [38]. presented a multiplier architecture with an approximate compressor unit that replaces the accumulation unit and a tool that helps the user find a balance between minimal errors, area, and power. Neural network applications demand high accuracy but also high energy consumption and resource utilization. Works such as those described,[6] use approximate multipliers to keep hardware resources and power consumption within acceptable limits. Lee et al[22]. introduced a run-time configurable Unified Neural Processing Unit (UNPU) with variable bit-precision that can perform and operations using different bit widths for weights.

## 7. Conclusion

One of the most famous computer vision tasks is object classification. Today, due to the increase in the number of CCTV in the cities, a lot of image data is produced, which is very important to process them, but due to the large size of image processing models, their high energy consumption and the large volume of image data, it is impossible to process this information in the old style , send this information to central server and process them there. Therefore, it is very important to design low-power and small model size of image processing that can run on edge devices that have limited computation resource. In this article we have explained our designed model named CluFy. CluFy is a low-power object classification model. CluFy can first make an initial estimate of the input image class by using Clu model in clustering layer and limit the number of classes participating in the classification layer. By doing this limitation, the size of the classification model, which is called Fy, is reduced (memory usage is reduced), which naturally reduces the number of parameters and the energy consumption. CluFy is much smaller than Yolo and GoogleNet in terms of model size and the number of operations, but due to its pipeline structure, it propagates errors and despite of having the $\alpha$ parameter that was explained in section 4.3, It is less accurate than those models, which in our future works, we will improve its accuracy by making changes in CluFy.

## References

[1] Stephanie B Baker, Wei Xiang, and Ian Atkinson. Internet of things for smart healthcare: Technologies, challenges, and opportunities. *Ieee Access*, 5:26521–26544, 2017.

[2] Sina Boroumand, Hadi P Afshar, Philip Brisk, and Siamak Mohammadi. Exploration of approximate multipliers design space using carry propagation free compressors. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 611–616. IEEE, 2018.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[4] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.

[5] Michael F Cowlishaw. Decimal floating-point: Algorism for computers. In *Proceedings 2003 16th IEEE Symposium on Computer Arithmetic*, pages 104–111. IEEE, 2003.

[6] GA Gillani, Muhammad Abdullah Hanif, Bart Verstoep, Sabih H Gerez, Muhammad Shafique, and Andre BJ Kokkeler. Macish: Designing approximate mac accelerators with internal-self-healing. *IEEE Access*, 7:77142–77160, 2019.

[7] Stephen Gould, Tianshi Gao, and Daphne Koller. Region-based segmentation and object detection. *Advances in neural information processing systems*, 22, 2009.

[8] Bruce Greer, John Harrison, Greg Henry, Wei Li, and Peter Tang. Scientific computing on the itanium™ processor. In *SC'01: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*, pages 1–1. IEEE, 2001.

[9] Soheil Hashemi, R Iris Bahar, and Sherief Reda. Drum: A dynamic range unbiased multiplier for approximate applications. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 418–425. IEEE, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] Wei Hu and Huanhao Li. A blockchain-based secure transaction model for distributed energy in industrial internet of things. *Alexandria Engineering Journal*, 60(1):491–500, 2021.

[13] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[14] Mohsen Imani, Daniel Peroni, and Tajana Rosing. Cfpu: Configurable floating point multiplier for energy-efficient computing. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2017.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[16] Min Soo Kim, Alberto Antonio Del Barrio Garcia, Hyunjin Kim, and Nader Bagherzadeh. The effects of approximate multiplication on convolutional neural networks. *IEEE Transactions on Emerging Topics in Computing*, 2021.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[18] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In *2011 24th Internatioal Conference on VLSI Design*, pages 346–351. IEEE, 2011.

[19] Ashok Kumar. The hp pa-8000 risc cpu. *IEEE Micro*, 17(2):27–32, 1997.

[20] Anil Kumar, Yuqing Zhou, CP Gandhi, Rajesh Kumar, and Jiawei Xiang. Bearing defect size assessment using wavelet transform based deep convolutional neural network (dcnn). *Alexandria Engineering Journal*, 59(2):999–1012, 2020.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[22] Jinmook Lee, Changhyeon Kim, Sanghoon Kang, Dongjoo Shin, Sangyeob Kim, and Hoi-Jun Yoo. Unpu: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 218–220. IEEE, 2018.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[24] Asuka Maki, Daisuke Miyashita, Kengo Nakata, Fumihiko Tachibana, Tomoya Suzuki, and Jun Deguchi. Fpga-based cnn processor with filter-wise-optimized bit precision. In *2018 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 47–50. IEEE, 2018.

[25] Shaik Nasar and K Subbarao. Design and implementation of mac unit using reversible logic. *International Journal of Engineering Research and Applications*, 2(5):1848–1855, 2012.

[26] Moustafa M Nasralla, Chaminda TER Hewage, and Maria G Martini. Subjective and objective evaluation and packet loss modeling for 3d video transmission over lte networks. In *2014 international conference on telecommunications and multimedia (temu)*, pages 254–259. IEEE, 2014.

[27] Moustafa M Nasralla, Nabeel Khan, and Maria G Martini. Content-aware downlink scheduling for lte wireless systems: A survey and performance comparison of key approaches. *Computer Communications*, 130:78–100, 2018.

[28] Moustafa M Nasralla, Manzoor Razaak, Ikram Rehman, and Maria G Martini. A comparative performance evaluation of the hevc standard with its predecessor h. 264/avc for medical videos over 4g and beyond wireless networks. In *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, pages 50–54. IEEE, 2018.

[29] Frank P O'Connell and Steven W White. Power3: The next generation of powerpc processors. *IBM Journal of Research and Development*, 44(6):873–884, 2000.

[30] Amiya Prakash and K Sharma. Design and analysis of multiplier accumulation unit by using hybrid adder. *J. Comput. Trends Technol*, 37(2):96–102, 2016.

[31] Sachin Raghav and Rinkesh Mittal. Implementation of fast and efficient mac unit on fpga. *International Journal of Mathematical Sciences and Computing (IJMSC)*, 2(4):24–33, 2016.

[32] Ikram U Rehman, Moustafa M Nasralla, and Nada Y Philip. Multilayer perceptron neural network-based qos-aware, content-aware and device-aware qoe prediction model: A proposed prediction model for medical ultrasound streaming over small cell networks. *Electronics*, 8(2):194, 2019.

[33] Arch D Robison. N-bit unsigned division via n-bit multiply-add. In *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*, pages 131–139. IEEE, 2005.

[34] Rodina Samy, Hossam Ali Hassan Fahmy, Tarek Eldeeb, Ramy Raafat, Yasmeen Farouk, Mostafa Elkhouly, and Amira Mohamed. Decimal floating-point fused multiply-add unit, Apr. 8 2014. US Patent 8,694,572.

[35] Shaik Masthan Sharif and D Prasad. Design of optimized 64 bit mac unit for dsp applications. *Int. Journal of Advanced Trends in Computer Science and Engineering*, 3(5):456–460, 2014.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[38] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. Tosam: An energy-efficient truncation-and rounding-based scalable approximate multiplier. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(5):1161–1173, 2019.

[39] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[40] Christian Wojek, Gyuri Dorkó, André Schulz, and Bernt Schiele. Sliding-windows for rapid object class localization: A parallel technique. In *Joint Pattern Recognition Symposium*, pages 71–81. Springer, 2008.

[41] Kehe Wu, Rui Cheng, Wenchao Cui, and Wei Li. A lightweight sm2-based security authentication scheme for smart grids. *Alexandria Engineering Journal*, 60(1):435–446, 2021.

[42] Zhenheng Yang and Ramakant Nevatia. A multi-scale cascade fully convolutional network face detector. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 633–638. IEEE, 2016.

[43] Chunsheng Zhu, Lei Shu, Victor CM Leung, Song Guo, Yan Zhang, and Laurence T Yang. Secure multimedia big data in trust-assisted sensor-cloud for smart city. *IEEE Communications Magazine*, 55(12):24–30, 2017.