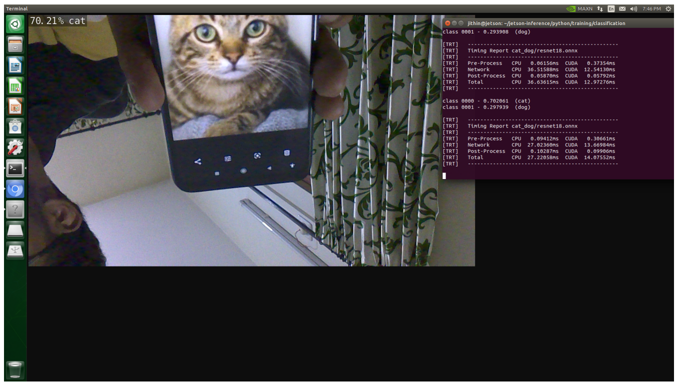
CUSTOM IMAGE CLASSIFIER

Amrutha KV, Jithin KM, Jithin T Mathews, Prasila P
B Tech Scholars, Dept. of CSE
Govt. Engg. College Wayanad

***Abstract*—Users are sharing a vast amount of data through apps every day. The large volume of digital data is being used by companies to deliver better and smarter services to the people accessing it. Image recognition refers to technologies that identify places, logos, people, objects, buildings, and various other variables in images. Image recognition is a part of computer vision and a process to identify and detect an object or an attribute in a digital video or image. However, the vast amount of data can be recognized and classified only with the use of a generally trained Convolutional neural network (CNN). There is no specific purpose trained CNN for recognizing data with custom requirements. This can be solved by creating a custom image classifier with transfer learning to identify specific classes of objects using Nvidia jetson nano and Pytorch. A Convolutional neural network is used to recognize the images accurately. It is an algorithm used for analyzing images and assigning importance to specific bits. The last few layers of CNN are retrained to recognize, identify, and classify images more deeply.**

***Index Terms*—CNN, ResNet**


70.21% cat
MAXN
7:46 PM
jithin@jetson: ~/jetson-inference/python/training/classification

```
class 0001 - 0.293908  (dog)

[TRT]   ---------------------------------------------
[TRT]   Timing Report cat_dog/resnet18.onnx
[TRT]   ---------------------------------------------
[TRT]   Pre-Process   CPU   0.06156ms  CUDA   0.37354ms
[TRT]   Network       CPU  36.51588ms  CUDA  12.54130ms
[TRT]   Post-Process  CPU   0.05870ms  CUDA   0.05792ms
[TRT]   Total         CPU  36.63615ms  CUDA  12.97276ms
[TRT]   ---------------------------------------------

class 0000 - 0.702061  (cat)
class 0001 - 0.297939  (dog)

[TRT]   ---------------------------------------------
[TRT]   Timing Report cat_dog/resnet18.onnx
[TRT]   ---------------------------------------------
[TRT]   Pre-Process   CPU   0.09412ms  CUDA   0.30661ms
[TRT]   Network       CPU  27.02360ms  CUDA  13.66984ms
[TRT]   Post-Process  CPU   0.10287ms  CUDA   0.09906ms
[TRT]   Total         CPU  27.22058ms  CUDA  14.07552ms
[TRT]   ---------------------------------------------
```

Figure 1. An example of identyfying cat along with confidence score [Example output from our project]

I. Introduction

Image Recognition and identification is a classic machine learning problem. It is a very challenging task to detect an object or to recognize an image from a digital image or a video. Image Recognition has applications in the various field of computer vision, some of which include facial recognition, biometric systems many more. Deep Learning algorithms have achieved great progress in the field of computer vision. Convolutional neural networks (CNN) are deep learning algorithms that have high accuracy and can train large data sets with millions of parameters, in form of 2D images as input and combine it with filters to produce the desired outputs. In this project, CNN models are built to evaluate its performance on image recognition and detection data sets. The algorithm is implemented using Nvidia jetson nano and PyTorch.

Custom image classifier will recognize something specific, for example identifying an automatic pet door for a specific breed of dog, or a plant species for sorting, or any other exiting applications you can think about. ResNet(residual neural network) CNN is used to train the model using python based on custom image requirements. And then it is implemented on an Nvidia jetson nano go board which is a RISC AI Module that can be used for image classification. The system will provide a real-time stream to the laptop along with classification output and confidence score. Thus the proposed system was able to identify more classes than the current system could do and was more precise too.

Figure 1 shows an example of image identification of the dog-cat model. This figure describes the accuracy of identification based object detection.

II. Materials and Methods

A. Dataset

A total of 5000 data set was collected for the cat-dog model, 10000 for different ten species of plants for plant model and 2000 for a custom model. Image Batch assistant was the tool used to collect data. It is an Ad-on that can be installed on your browsers. Useful for downloading all the images found on a web page as a batch file rather than downloading them individually.

B. Experimental setting

The collected data need to be set up before it can be fed as input for training. The dataset was randomly split into three separate sets, 70% for training and 20% for validation, and 10% for testing. The training set which contained the majority of data was used to train our model while the validation set was used to estimate how well our model had been trained. Finally, the test set was used by the user to test the performance of the constructed model.

To avoid noise and perform pre-processing Feature extraction technique was performed, where it tries to highlight the important parts of the image. However, doing that on a data set that contains more than 10,000 images manually was thought to be infeasible. Deep learning was the second option we attempted but these could only be used to separate human-like faces efficiently. Therefore it could not be done on the data set of pens, plants.

For further accuracy, Hyperparameter learning was tried out. Here we manually modify the layers in the neural network and make slight variations in parameters and see how this affects the model accuracy. Minor improvements were made during hyperparameter learning for the custom dataset owning to poor training data. Notable improvements were found in the plant dataset.

C. *Transfer Learning*

In general, building a classification model from scratch takes a lot of time and a huge amount of data set. These would have to be trained on high-performance GPUs for months to achieve a significant accuracy level. Transfer learning is the machine learning technique where a model trained on one task is re-purposed on a second related task. Transfer learning relates to problems such as multi-task learning and concept drift and is not exclusively an area of study for deep learning. Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging data sets on which deep learning models are trained. Transfer learning mainly works in deep learning if the model features learned from the first task are general. [1]

D. *Transfer Learning Approaches*

Develop Model Approach [1]

- Select Source Task. You must select a related predictive modelling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.
- Develop Source Model. Next, you must develop a skilful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.
- Reuse Model. The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modelling technique used.
- Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

Pre-trained Model Approach [1]

- Select Source Model. A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
- Reuse Model. The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modelling technique used.
- Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

E. *Developer Kit Architecture*

Figure 2 is the top view of NVIDIA Jetson Nano. NVIDIA Jetson Nano is a small, powerful computer that lets us run multiple neural networks in parallel for applications like image classification, object detection. Jetson Nano is a GPU-enabled edge computing platform for AI and deep learning applications. The GPU-powered platform is capable of training models and deploying online learning models and is most suited for deploying pre-trained AI models for real-time high-performance inference. The neural network model supported by NVIDIA Jetson Nano is ONNX (Open Neural Network Exchange) model.ONNX supports interoperability between frameworks, which means we can train a model in one of the many popular machine learning frameworks like PyTorch which we have used in our project, and convert it into ONNX format and consume the ONNX model. The NVIDIA Jetson Nano module features a 1.43 GHz ARM Cortex-A57 quad-core processor and 128-core NVIDIA Maxwell graphics with speeds up to 921 MHz. Jetson Nano delivers 472 GFLOPs for running modern AI algorithms fast. The camera along with it has MIPI CSI-2 DPHY lanes.

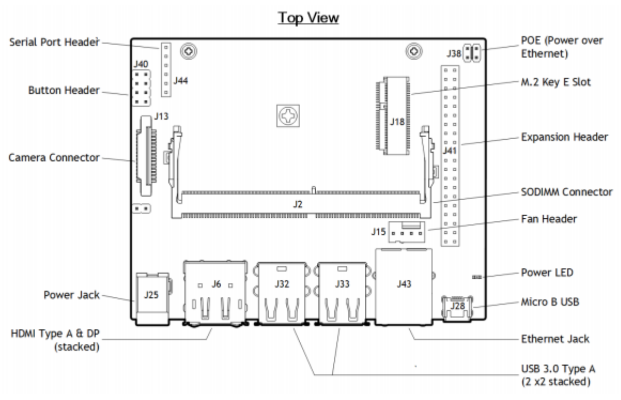


Figure 2. Top view of NVIDIA Jetson Nano

F. Process Model

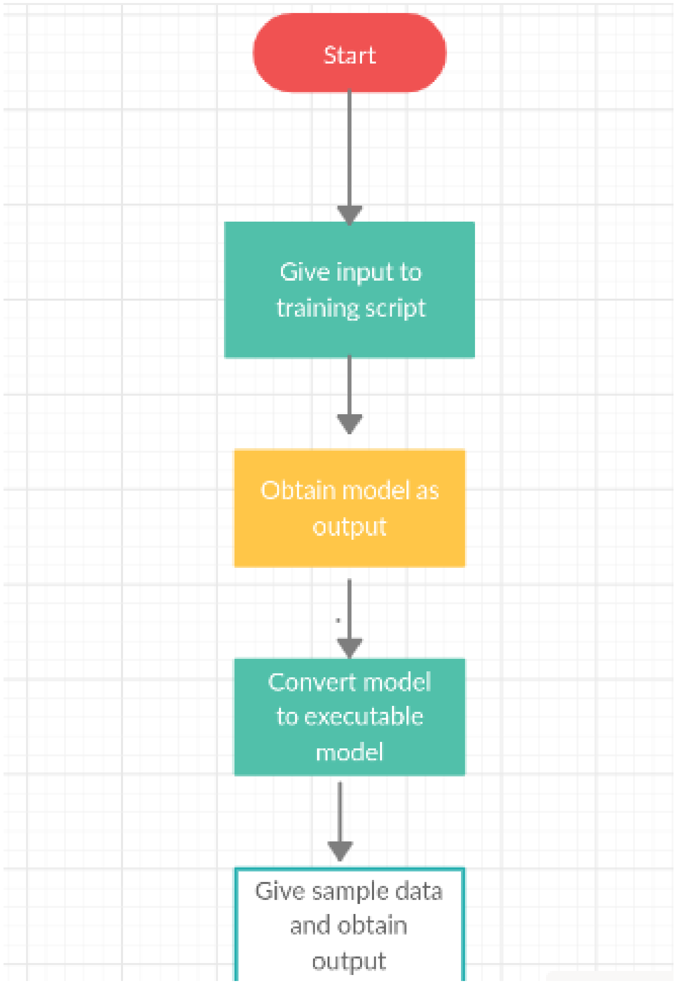

Start
Give input to training script
Obtain model as output
Convert model to executable model
Give sample data and obtain output

Figure 3. Flow chart showing the basic steps of the Process cycle

Figure 3 shows an abstract view of the system training cycle. This has to be done until the output is of acceptable performance. The initial process starts with the collection of data. The data in the training set are then fed as input into the training script to train the model. Once training is completed a PyTorch model is obtained. The accuracy of the model is verified by sanity checks by providing data in the validation set. This validation set will be used for testing the model accuracy after each epoch/training rounds. This model then has to be fed into the Nvidia jetson nano development kit. Since the Nvidia jetson nano accepts only ONNX (Open Neural Network Exchange) model, the Pytorch model is converted to the ONNX model and fed into the development kid. The ONNX model is the final model for classification and identification. Data in the test set is finally used by the user to test the accuracy of the constructed model.

G. Confidence Score

Classification confidence scores are designed to measure the accuracy of the model when predicting class assignment (rather than the uncertainty inherent in the data). Most generative classification models are probabilistic, and therefore provide such confidence scores directly. Most discriminative models, on the other hand, do not have direct access to the probability of each prediction. Instead, related non-probabilistic scores are used as proxies, for example, the margin in SVM classifiers [2]. SVMs work by determining a separating hyper-plane between the two classes. For a new test sample, it classifies it as class 1 or 2 according to which classes it is closest to. The decision-function(X) gives you the distance of the sample to the hyper-plane, or the distance to each of the two classes. This distance is a value that can be too large or too small according to where your sample is located from that hyper-plane or from the different classes.

$$CS(i) = \frac{100 - d(i)}{\sum_{n=0}^{n} = Nd} \tag{1}$$

Where CS(i) is the confidence score of that instance, i is an instance,d is the distance of an instance to a specific class,N is the number of classes in the model.

III. Related Works

In recent years there have been great strides in building classifiers for image detection and recognition on various data sets using various machine learning algorithms. Deep learning, in particular, has shown improvement in the accuracy of various data sets. We have gone through some of the works on image classification and studied their features, properties, and implementation details. The different aspects we have gone through are image recognition using machine learning and histogram.

a) : P. Viola and M. Jones, in their paper "Rapid object detection using a boosted cascade of simple features," [3] describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the "integral image" which allows the features used by the detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers. The third contribution is a method for combining increasingly more complex classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. The cascade can be viewed as an object-specific focus-of-attention mechanism which unlike previous approaches provides statistical guarantees that discarded regions are unlikely to contain the object of interest. In the domain of face detection, the system yields detection rates comparable to the best previous systems.

Used in real-time applications, the detector runs at 15 frames per second without resorting to image differencing or skin colour detection. This was totally limited to the facial detection of human beings. Another thing was it can't handle large data set and the whole process was time-consuming too.

b) : Navneet.Dalal and Bill. Triggs in paper "Histograms of Oriented Gradients for Human Detection" [4] studied the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient-based descriptors, they showed experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. The paper says about the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so they introduced a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds. This was limited to identify human detection. It uses only a continuous data set and even does not provide specific classification.

IV. Performance Evaluation

The performance of the proposed method is analysed by comparing its accuracy with a variation in the number of epochs. As seen in 4 change in confidence score is identified with a change in the number of epochs. In 4 the X-axis represents the number of epochs and the Y-axis represents accuracy. There are two lines shown, one for top 1 accuracy while other for top 5 accuracies. The top-1 accuracy rate is the ratio of images whose ground truth category is exactly the prediction category with maximum probability, while the top-5 accuracy rate is the ratio of images whose ground-truth category is within the top-5 prediction categories. The proposed system will provide real-time classification along with its confidence score. The model was able to classify the plants with good confidence score but the score for the custom dataset was relatively low, this was because of the lack of training data and the data that could be collected was of low quality. it was found out that the accuracy of the model depends on :

- The size of training data up to a point.More the data, the better the accuracy.
- The number of training rounds or Epochs done. On average 35 epochs produce good results and accuracy starts to converge at around 70 epochs.
- The number of input layers used. More layers provide better top class predictions.
- Hyperparameter learning where we manually reshape the neural network was able to provide an accuracy boost of 2-10 percent provided the training data was sufficiently large.

For further accuracy, Hyperparameter learning was tried out. It manually modifies the layers in the neural network and make slight variations in parameters and see how this affects the model accuracy. Minor improvements were made during hyperparameter learning for the custom dataset owning to poor training data. Notable improvements were found in the plant dataset. Next to avoid noise and perform pre-processing, feature extraction where we try to highlight the important parts of the image were tried out. However, doing that on a data set that contains more than 10,000 images manually was thought to be infeasible.

Epoch	Accuracy of Top-1	Accuracy of Top-5
0	10	42
10	19.68	53
20	24.5	60.38
30	30	66.28
40	40.03	75
50	42	77.5
60	50.2	80
70	55	85

Table I
Accuracy variation for plant model.

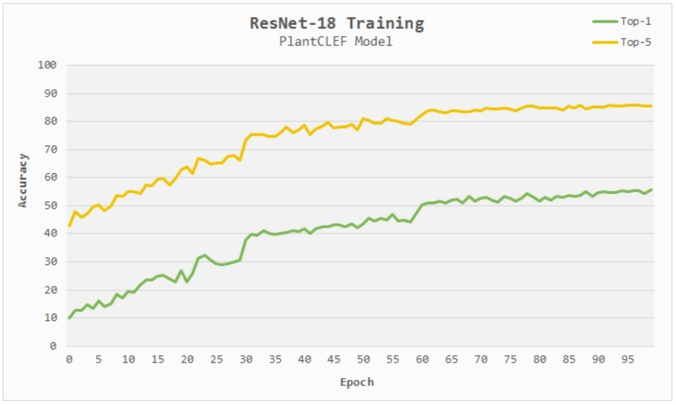


Figure 4. Comparision chart of plant model

V. Limitations

Since there are a lot of image classes that can be more deeply and specifically classified, it would be impossible to create a perfect one for all models that can classify everything it is given as input. Therefore we have to choose things the model will classify and train the model on those sample images. A new model will have to be constructed for each instance of new user demand and that will be time consuming and training will have to be done from the beginning. Sometimes the accuracy of the model is deeply affected by the quality of the camera attached to the development board. The camera attached to the development board had poor autofocus performance and its quality was not much satisfying. Another limitation was that the model may be quick to detect the noise in the test data image's environment and classify it accordingly, due it this reason the model may sometimes classify based on the background data. A large amount of data was required to obtain enough accuracy and that was the major limitation we faced in our project due to the lack of availability of data set for pen, marker, etc. This made the project hard because data size is important in machine learning. Another aspect that was not considered in the project was that the output obtained after classification was not utilized or was not converted to any mechanical action for example raising an alarm if it identified a snake.

VI. Conclusion

With today's technology, image recognition is becoming more and more reliable. The success rate is high as it can deal with 2D and well as 3D images. The image recognition using a convolutional neural network can fully automate the process and ensure its accuracy at a very high rate and can even train large data sets with millions of parameters. This says about higher convenience and lowers costs. By retraining the last few layers of the convolutional neural network precise classification with better accuracy is possible. As a result, can identify more classes than the original model. Their ability to learn by example makes them very flexible and powerful. They are also very well suited for real-time systems because of their fast response and computational times which are due to their parallel architecture. Another advantage is that system can be integrated into machines so that it can do a specific action once a specific class is recognized, for example, opening the dog house if it's your pet dog and not for any stray animal. The GPU-powered platform in Nvidia Jetson Nano had the capability of training models and deploying online learning models and is most suited for deploying pre-trained AI models for real-time high-performance inference. Image recognition is a powerful technology that brings immense advantage to the companies and end-users, helps them enhance their security, and track down the trespassers.

References

[1] https://machinelearningmastery.com/transfer-learning-for-deep-learning/,2019.

[2] https://www.arxiv-vanity.com/papers/1709.09844,2016.

[3] P. V. M. Jones, "Rapid object detection using a boosted cascade of simple features," *Mitsubishi Electric Research Labs Compaq CRL 201 Broadway, 8th FL One Cambridge Center*, vol. 14, no. 2, pp. 131–142, 2001.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *INRIA Rhone-Alps*, 2005.