# Alpha ML

P.S.S ABISHEK
Computer Science and Engineering Department
*(of Affiliation)*
SRM Institute of Science and Technology
Chennai, India

SOWRABH S
Computer Science and Engineering Department
*(of Affiliation)*
SRM Institute of Science and Technology
Chennai, India

S. SRIDHAR
ASSISTANT PROFESSOR
Computer Science and Engineering Department
*(of Affiliation)*
SRM Institute of Science and Technology
Chennai, India

*Abstract*—Getting started with your own machine learning projects might seem intimidating, but Alpha ML is a web-based tool that makes it fast, easy, and accessible to everyone. For instance, Alpha ML let anyone teach their computer to recognize images using a webcam. This will probably be the first time for a lot of people, to experiencing what it's like to train their own machine learning model: teaching the computer how to recognize patterns in data (images, in this case) and assign new data to categories.
Alpha ML lets you train your own machine learning model with the click of a button, no coding required, and export it to websites, apps, physical machines and
more.
You can use Alpha ML for Image recognition. Upload your own image files or capture them live with a webcam. These examples stay on-device, never leaving your computer unless you choose to choose to export it as an API. Extreme customizability allows you to tweak parameters and see how it works under the hood.

Additionally, with the use of Transfer learning [1] algorithm we propose to implement and optimize training new models while maintaining performance on existing ones.

***Keywords—Transfer Learning, Image classification, Mobile Net v2***

## I. INTRODUCTION

People are using AI to explore all kinds of ideas—identifying the roots of bad traffic in Los Angeles, improving recycling rates in Singapore, and even experimenting with dance. Machine Learning, an application of artificial intelligence (AI), gives the ability for a system to automatically learn and improve from experience without being explicitly programmed. Access to data remains the essential requirement for this to happen. Hence, the process of learning begins with observations or data. In this application we demonstrate one of the applications of machine learning called Image classification which refers to a process in compute vision that can classify and image according to its visual content.
Humans have an inherent ability to transfer knowledge across tasks. What we acquire as knowledge while learning about one task, we utilize in the same way to solve related tasks. The more related the tasks, the easier it is for us to transfer, or cross-utilize our knowledge. Conventional machine learning and deep learning algorithms, so far, have been traditionally designed to work in isolation. These algorithms are trained to solve specific tasks. The modelsmust be rebuilt from scratch once the feature-space distribution changes.With the help of Transfer learning [2], a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. The idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones.It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

## II. LITERATURE REVIEW

The groundwork for the project revolves around understanding the needs of people who want to learn Machine learning and its application. Although there are lot of products like this available in the market the result from the marketresearch, we did help us understand that the products are not readily available to consume for any lay man user. Having an edge over other products give us the ability to reach a number user base.

## III. METHODOLOGY

This section defines the architecture defined and the corresponding components that are a part of the application.

### *A. Architecture Diagram*

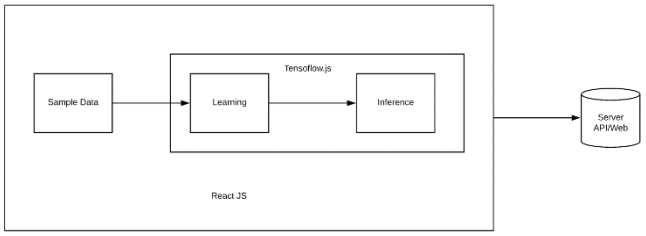


**Figure 1:** Model Summary

An overview of our work is to essentially build a system that provides an interface which can be used with ease to perform image classification. Using React.js we have implemented the meticulously designed interface keeping user experience in mind.

For training the model we use JavaScript version of TensorFlow [3], that has been popular in the market quite some time. The web application is hosted on web server with necessary node packages installed for this application.

### B. *Sample Data*

This component allows user to add sample data that can be taken live using the device's webcam or upload existing data from local system. For image classification per say user can add label to every class. Our primary goal is to keep user data at maximum privacy. This is achieved by storing all sample data on the browser and training model in their local machine. In this way no data is uploaded to the server and user are not worried about privacy of their datasets.

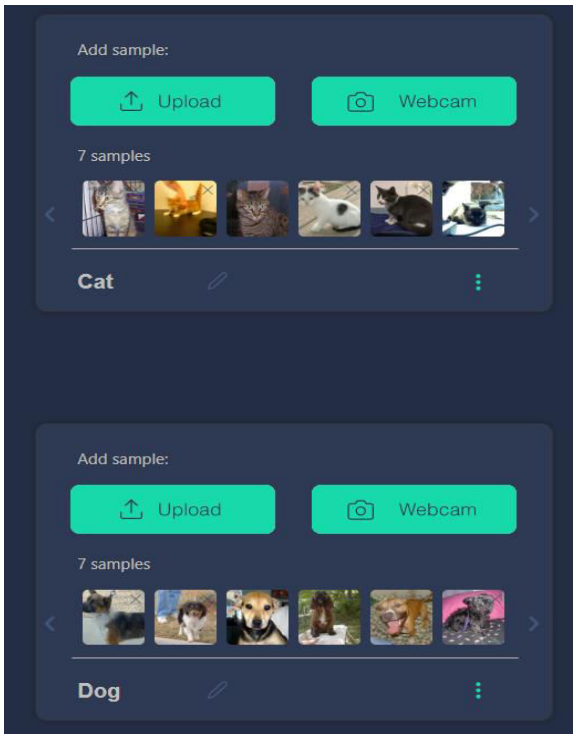


**Figure 2:** Adding data to classes

### C. *Learning*

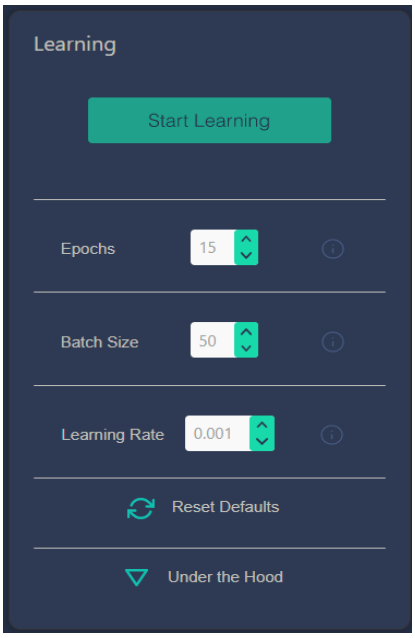


**Figure 3:** Learning Component

This component is specifically meant for tuning hyper parameters for training the model on the user clicking start learning. The following are the hyper parameters,

*a) Epochs: In terms of artificial neural networks, an epoch refers to one cycle passing the training dataset to train the model. The number epochs greatly depend on the accuracy of the neural network. Usually, training a neural network model takes more than few epochs. Although, there is no guarantee that the neural network will get better by increasing the epochs significantly.*

*b) Batch Size: Batch size refers to the number of samples to deliver to the model per iteration. This controls the stability of the model wherein feeding larger set of samples means the model makes very large gradient updates and very small gradient updates.*

*c) Learning Rate: This hyperparameter that controls how much need to change in order for the model to respond to the estimated error each time the model weights are updated. It is also called as Step size. This configurable hyperparameter can be of a very small positive value ranging from 0.001 to 1.0. Learning rate also helps how quickly the model can be updated to the problem.*

When the user clicks start learning these hyperparameters are forwarded to the model to training as an argument.

### D. *Inference*

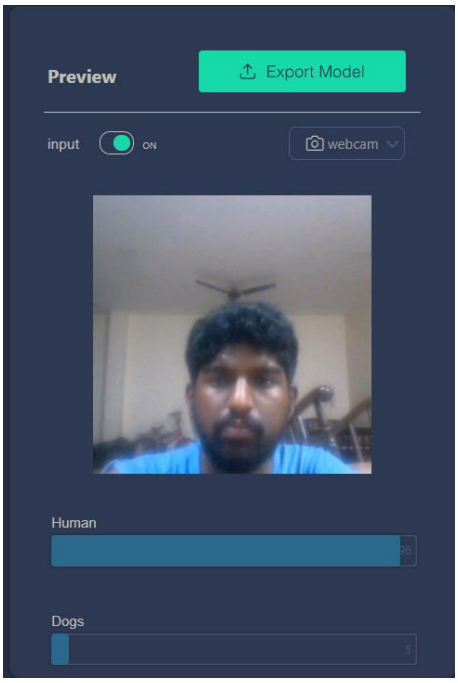


**Figure 4:** Inference Component

Upon training we can check the model for the given input we can understand to which class it belongs to. Like sample data module we can import live images from webcam or upload from file system. In case of webcam tensors are generated for every frame per second and is fed to the model to predict the probability of which class the sample belongs to. Hence the prediction bar fluctuates in real time.

### *E. Traning Model*

We are using MobileNet v2 [4] as our image classifier.MobileNets are small, low-latency, low-power models parameterised to meet the resource constraints of a variety of use cases. According to the research paper, MobileNet v2 improves the state-of-the-art performance of mobile models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. It is a very effective feature extractor for object detection and segmentation. For instance, for detection, when paired with Single Shot Detector Lite, MobileNet v2 is about 35 percent faster with the same accuracy than MobileNet v1 [5]. Before we start training the model, we download MobileNet v2 that is hosted in our server to train the model locally. Once downloaded the model we snip out the last layer that will be the prediction layer and replace it with our newly defined prediction layer based on the number classes. Once we model ready as per our requirement, we trained the model with the training data to determine which images belongs to which class.

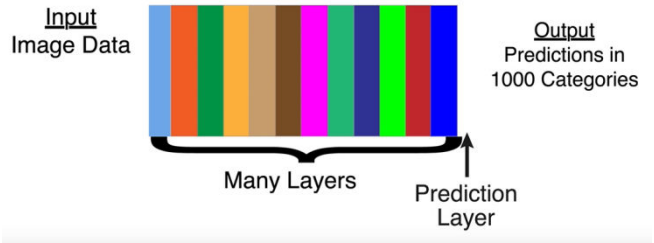


**Figure 5:** Snipping last layer

## IV. RESULT

Implementation of transfer learning helped improve the accuracy and training time significantly. For an uploaded image of a dog we received a 98% probability of the image belonging to the Dog class. Hence, we can infer from the result that the application has executed successfully.

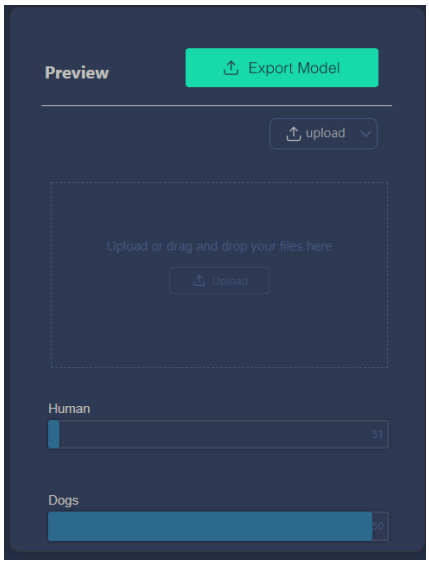


**Figure 6:** Probability of dog image

### *A. Optimization*

```
const load_model = async () => {
  const model = await tf.loadLayersModel("https://alpha-model.herokuapp.com/download");
  console.log("model is loaded");
  await model.save('indexeddb://base-model');
  console.log("model is saved to indexDB");

  model.dispose();
}
```

**Figure 7:** Loading model only once

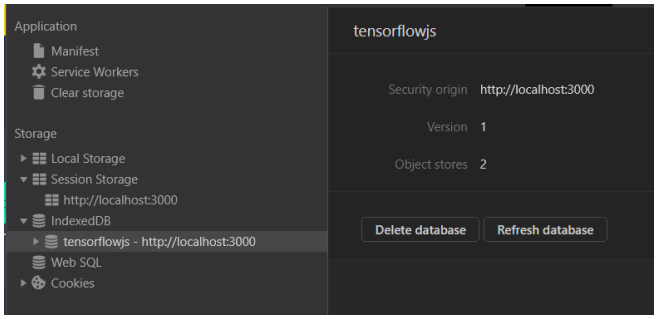


**Figure 8:** Saving model to IndexDB

There is certain optimization we intend to do regarding the way on how we train our model. Instead of downloading the pre trained model from the server by calling the API every time we stored it in browser's IndexDB which provides us with enough storage to store our model. Additionally, we in order to be memory efficient we dispose any duplicate copy of tensors that are not in use.

### *B. Limitation*

On running the application on mobile version of browser we met high memory usage eventually leading the application to crash and mobile to reboot.

Additionally, it is worth pointing out that Transfer Learning operates on distincttasks. Like many multitask learning methods, it cannotproperly deal with domains that are continually changingon a spectrum (e.g., old task being classification from topdownview, and new task being classification from views ofunknown angles); the tasks must be enumerated. In addition, transfer learning requires each sample to be accompanied by theinformation of which task it belongs to, and this informationis needed for both training and testing.

The application contains lot of conditional rendering of components due to same usage of interface for both mobile and desktop version. This affects the performance in some scenarios.

## V. CONCLUSION

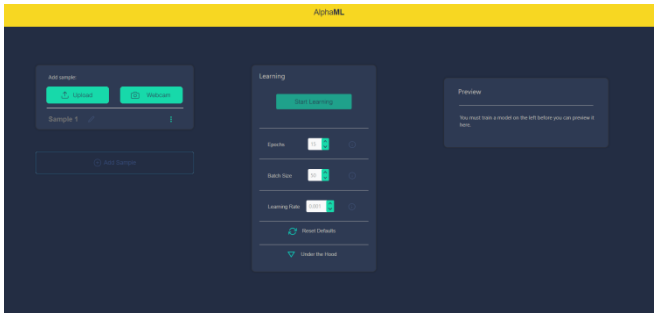


**Figure 9:** Alpha ML Landing Page

The result obtained from our proposed work helps user perform image classification without any prior knowledge on machine learning or coding. Our model was tested on Mobile Net v2 to perform transfer learning that gave interesting result in terms of accuracy and training time.

### *A. Future work*

The following are the modules and upgrades that are expected to be completed at a later stage. At this point our work has not implemented any other applications of machine learning like object detection, facial recognition. So, we propose to implement these modules soon.

Additionally,if the device does not have enough resources to train the model, the data obtained for each class can be uploaded to a private server and the model can be trained remotely.

## VI. REFERENCES

[1] Z. Li and D. Hoiem, "Learning without Forgetting," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 12, pp. 2935-2947, 1 Dec. 2018.

[2] L. Shao, F. Zhu and X. Li, "Transfer Learning for Visual Categorization: A Survey," in IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 5, pp. 1019-1034, May 2015.

[3] C. Li and 7C. Li, "Web Front-End Realtime Face Recognition Based on TFJS," 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Suzhou, China, 2019, pp. 1-5.

[4] D. Sinha and M. El-Sharkawy, "Thin MobileNet: An Enhanced MobileNet Architecture," *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York City, NY, USA, 2019, pp. 0280-0285.

[5] D. Sinha and M. El-Sharkawy, "Ultra-thin MobileNet," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0234-0240.

Hands-on Guide to Transfer Learning 2015 - https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

File Drag and Drop - https://react-dropzone.js.org/

Tensoflow.js - https://js.tensorflow.org/api/latest/