Image Classification: Human Detection and Classification

Sydney Blackburn
*Red ID:*816557736
Email ID: sblackburn@sdsu.edu

Atul Doki
Red ID: 824400246
Email ID: adoki75312sdsu.edu

Ajinkya Deshmukh
Red ID: 824662612
Email ID: adeshmukh6432@sdsu.edu

***Abstract*— In the fields of Computer Vision and Machine Learning, the importance of image classification cannot be understated. Specifically, the use of image classification models such as CNN, KNN and SVM are the core engines behind the proper functioning image classification. A long research is being done in the field of image classification and due the results of this research, it's possible in this date to develop production grade technologies with low error rate and high accuracy level. The computational power has increased significantly over the years and along with it, the quality of pattern recognition systems has also gone up. In this proposal, we discuss the comparison of three image classification algorithms – Convolution Neural Network (CNN), K-Nearest-Neighbors (KNN) and Support Vector Machines (SVM). We will discuss the properties and characteristics of all these three models along with the pros and cons of each of these and finally arrive at a conclusion on which one has the highest performance. We will consider learning curve and accuracy as the performance measures for these models and compare these measures on the CIFAR-100 dataset. In this proposal, we train our models to detect a human in an image and classify it into one of the 5 categories which are baby, boy, girl, man and, woman. It is observed that KNN has the highest accuracy of all the proposed models.**

***Index Terms*— Machine Learning, Convolution Neural Network (CNN), K-Nearest-Neighbors (KNN), Support Vector Machines (SVM).**

I. INTRODUCTION

Image classification is one of the most important parts of the analysis of digital images. The primary purpose of the classification process is to categorize all the pixels in a digital image into one of several classes. This categorized data may then be used to produce thematic maps of the land cover present in an image. These thematic maps are then used to classify the image. The ability to recognize and classify objects by humans is an important application of the power of human sight. Recognizing humans and classifying them based on numerous factors such as age, gender, skin-color, etc. is easy to the trained eyes, but challenging to automate. However due to the extensive research in the field of Computer Vision, there are numerous supervised and unsupervised classification models which can be applied to this problem. These models vary in the algorithm & accuracy and have their own pros and cons. Hence, selection of a classification model for a particular problem statement is a tedious task.

In our proposal, we explain mainly 3 supervised models Convolution Neural Network (CNN), k-Nearest Neighbor (KNN) and Support Vector Machines (SVM) which analyze images from dataset *CIFAR 100*. The intent here is to train the models with the 'train images dataset' and test the model on 'test images dataset' and compute the accuracies of each model. In this proposal, we train our models to detect a human and classify them into 5 categories which are baby, boy, girl, man and, woman. We will then compare the accuracies of each model.

II. DATASET & EXTRACTION

In this section, we will look at the dataset used and how the images were extracted

A. Dataset

The dataset used for this project is the CIFAR-100 dataset. This dataset contains 100 classes with 600 images each. Each class has 500 training images and 100 testing images. The 100 classes are grouped into 20 super classes consisting of 5 classes each [1]. We chose the "people" superclass which has the classes baby, boy, girl, man, and woman.

B. Data Extraction

To extract the data, we first get the file path to the dataset. Then, load the meta, training, and testing .mat files to get the labels and the images. The meta data holds information about the training and testing files, and the other two files contain a 2-D array of the red, green, and blue channels of each image and a list of numbers where the number at index i indicates the label of the ith image in the data array [1]. From the meta data, we find the index that corresponds to the "people" superclass. From here we can use the superclass index to find the indices within the training and testing data that match the "people" superclass. We use the same process to grab the classes from the superclass. We need to do this five times for each class.

Next, we manipulate the training and testing data. We reshape the 1-D image array information into a 4-D structure which holds the length, width, and channels of each image. We then permute the data into the correct order. In this case, we are just switching the columns and rows so the image is displayed correctly.

Finally, we store the data into training and testing folders. These folders include five other folders corresponding to the classes we are using to classify the images.

III. Convolution Neural Networks

In this section, we will look at how well Convolution Neural Networks classifies types of humans.

A. *Algorithm*

The convolution neural network model consists of an image input layer, three convolution layers, three batch normalization layers, three relu layers, two max pooling layers, a fully connected layer, a softmax layer, and a classification layer for a total of fifteen layers. The image input layer inputs 2-D images to a network and applies data normalization [7]. Here we specify the size of the image, including the length, width, and number of channels. Each image in our dataset is 32x32x3. The 2-D convolution layer applies sliding convolutional filters to the input [3]. We specify the filter size and number of filters. We set the filter size to 3 and the number of filters to 8. We can also specify name-value pair arguments, and in this case used "padding" and "same". The batch normalization layer normalizes each input channel across a mini-batch. This layer speeds up the training and reduces the sensitivity to network initialization [1]. The rely layer performs a threshold operation to each element of the input, where any value less than zero is set to zero [9]. The max pooling layer performs down-sampling to reduce spatial size of feature map and remove redundant spatial information [8]. We set the pool size to 2 with a stride of 2. The fully connected layer is where the neurons connect to all the neurons in the preceding layer. This layer multiplies the input by a weight matrix and then adds a bias vector [6]. Our output size is 5, and this corresponds to the five classes. The SoftMax layer normalizes the output of the fully connected layer [4]. Lastly, the classification layer computes the cross-entropy loss for multi-class classification problems with mutually exclusive classes [2].

We trained the network by setting the training options with stochastic gradient descent with momentum (sdgm), and initial learning rate of .01, max epochs of 4, and shuffling every epoch.

Finally, to compute the accuracy, we used the formula:

$$\cdot \sum \frac{Number\ Correct}{Total\ Number}$$

where number correct is the number of labels the network predicts correctly and total number is all the validation labels

B. *Experiments*

Table #.1 shows the results of different experiments used to increase the accuracy of the model. The initial values expressed in Algorithm section gives an accuracy of 33.8%. Increasing the pool size to [3 2] improved accuracy by .01. However, the speed decreased slightly.

Experiment	Accuracy	Speed
Initial Values	.3380	5m55s
Changed max pooling layer to have pool size of [3 2]	.3480	6m4s
Increase number of filters to 9	.3360	5m38s
Adding another set of layers	.3020	6m40s
Increased filter size to 8	.3120	6m50s
Decreased learning rate to .001	.3042	8m56s
Increased epochs to 10	.3540	15m7s
Increased epochs to 10 and adding a dropout layer with a probability of .02	.3560	14m49s
Increased epochs to 10, decreased learning rate to .001, and gradual increased dropout layer probability from .02 to .04 to .05	.2700	15m11s
Increased epochs to 10 and changed to rmsprops	.3500	15m7s

Table #.1. **Accuracy and Speed Results of different Convolution Neural Network Experiments**. The accuracy is expressed as a decimal where multiplying by 100 gives the accuracy as a percentage. The speed is expressed in minutes and seconds.

The next four experiments decreased the accuracy and decreased the speed. The accuracy increased again when increasing the max epochs to 10. Here the accuracy increased to .3540 and the speed stayed relatively the same. Figure #.1 shows the training progress of the experiments that increased the epochs to 10. We can see that the validation and training lines diverge meaning we are overfitting. To solve this, we added a dropout layer. The dropout layer randomly sets input elements to zero with a given probability [5]. This layer is useful when there aren't enough training images. In this case, we did not have enough training images, so the network was basically memorizing which label correctly classified the testing images. Adding this extra layer increased the accuracy by .002 and increased the speed by about a minute. This experiment ended up having the best results.

C. *Performance*

Convolution Neural Networks is not a good model for classifying types of humans. More training data could have helped increase the accuracy but only to an extent. The similarity of facial structures between the different types of humans makes it hard for Convolutional Neural Networks to differentiate the classified images from one another.

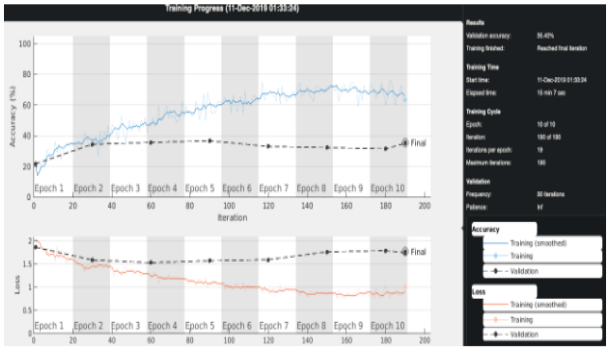

Figure #.1. **Training Progress for Increasing the Number of Epoch to 10**. The validation and the training lines diverge meaning we are overfitting.

IV. K-NEAREST-NEIGHBOR CLASSIFIER

KNN is a robust and versatile classifier that is used as a benchmark for other classifiers such as Artificial Neural Network(ANN), Support Vector Machines(SVM). Despite being simple it can outperform many other classifiers and it has a variety of applications such as classification, pattern recognition, gene recognition, check document similarity etc.

A. Introduction

KNN is a supervised learning algorithm which means we are given a set of training observations with their respective labels and would like to capture the relationship between data and the label so that it could classify easily any unknown data produced to the algorithm.

KNN classifier is also known as non-parametric and instant-based algorithm.

1. Non-parametric means that the model doesn't make any assumptions of the training data , avoiding the danger of mismodeling the data.
2. Instant-based learning means that the algorithm doesn't learn the training data explicitly instead it memorizes the data and uses this data to predict unknown data.

It is worth noting that the training phase of KNN comes at a memory cost as we have to store the huge training data sets computation cost as we have to go through entire training data set to predict the label of unknown data.

B. Working of KNN

In KNN model, k is the number of nearest neighbors which gives a majority voting to an unseen data. Similarity between unseen data and k nearest neighbor is determined by distance metrics. It assigns unseen data to that class to which majority of k nearest neighbors belong.

Let's see the working of this algorithm with the help of an example.
Suppose you have a dataset which has two classes Red circles (RC) and Green squares (GS) as plotted in the figure (1).

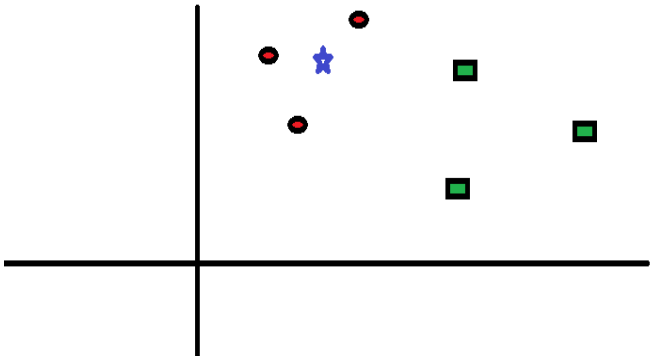

Fig. 1. classes of datasets presented by color

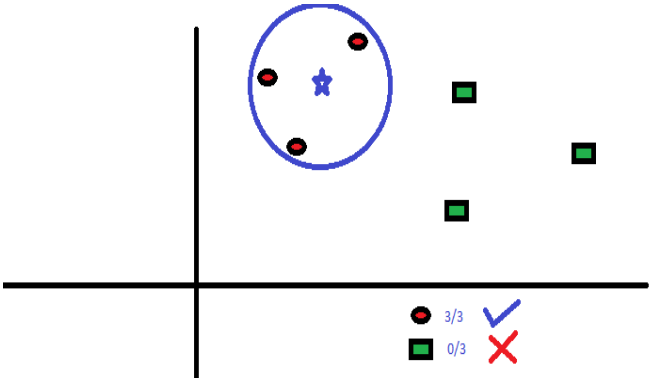

Fig. 2

Now we have to classify to which class the blur star belongs to. Suppose we take K=3 , then we check 3 nearest neighbors of the blue star using any distance metric. As you can see in figure (2) 3 closes neighbors to blue start are Red circles (RC) hence the blue star belongs to class Red circles (RC). The choice was obvious as all 3 votes from the closest neighbor wen to Red circles (RC).

C. Hyper-parameters

Hyper-parameters are those parameters that are predefined and different parameters can change the accuracy of the unknow data. KNN model include two hyper-parameters:

1. K values – k nearest neighbors around the unseen data
2. Distance metric – metric to find out the k nearest neighbors.

e.g. Euclidean distance $d(I,\ J\ ) = \sqrt{\sum_p (I_p - J_p)^2}$

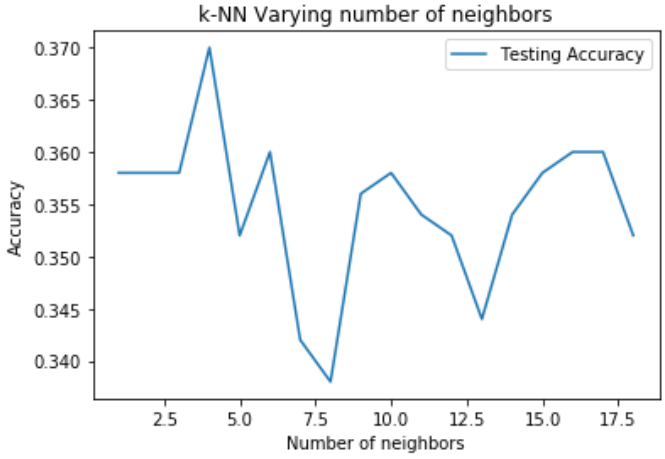

Fig. (3)

D. Classification of Human Using KNN

We will be using KNN model to classify if an image is a baby, boy, girl, man or woman. Initially we scale all the images to 32 x 32 x 3 to improve convergence speed and accuracy. We are going to use KNeighborsClassifier class from Scikit-Learn library to train the model with the training data and make predictions. First we trained our model with k = 3 nearest neighbors and we got an accuracy of 30.4. But we

tried to train our model with different k values and we were able to increase the accuracy of model to 31.8 with k=16 nearest neighbors using default distance metric Euclidean distance. When we used distance metric Manhattan distance to improve our model accuracy by 6% to 37% with k=4 nearest neighbors. As you can see in the graph below that the accuracy of the model decreased drastically after k=4.

E. Confusion Matrix

We have used confused confusion matrix to show number of data classified correctly and incorrectly which gives us some insight into the errors being made by the classifier.

```
array([[69,  8,  7, 15,  1],
       [35, 32,  9, 21,  3],
       [43, 17, 12, 18, 10],
       [26, 16,  9, 43,  6],
       [41,  9,  5, 22, 23]])
```

Fig. 4

F. Classification Report

We have used Classification report to show the quality of KNN model. As you can see in the figure (5) that our model was able to predict 53% of the women correctly and it could only predict 29% of girls correctly. The figure also shows other metrics like recall, f1-score which defines quality of our model.

	precision	recall	f1-score	support
0	0.32	0.69	0.44	100
1	0.39	0.32	0.35	100
2	0.29	0.12	0.17	100
3	0.36	0.43	0.39	100
4	0.53	0.23	0.32	100
accuracy			0.36	500
macro avg	0.38	0.36	0.33	500
weighted avg	0.38	0.36	0.33	500

Fig. 5

G. Improvements

1. As we have used 2500 images to train our model, we could use more data train our model in a better way to increase the accuracy.
2. Changing distance metrics like Chebyshev, WMinkowski, etc. can also improve the accuracy.
3. Dimensionality reduction techniques like PCA can be used before training our model will help make the distance metric more meaningful.
4. Approximate nearest neighbor techniques like using k-d trees to store training observations can be used to decrease testing time.

V. SUPPORT VECTOR MACHINE (SVM)

In Support Vector Machines (SVMs, also support vector networks) analyze data used for classification and regression analysis. The degree of confidence measures the probability of misclassification. So, we define terms functional margin and geometric margin. A functional margin is the accuracy of classification of a point whereas Geometric margin is the normalized version of functional margin. It computes the Euclidean distance between the hyperplane (or linear classifier) and the data points.

Suppose, we are given a set of training examples, each marked as belonging to one or the other of two categories. An SVM training algorithm works on a model that assigns new examples to one category or the other.

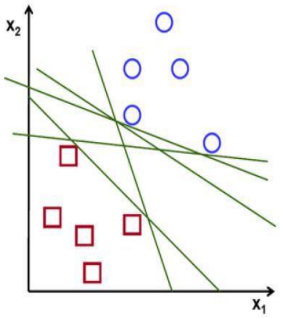

Fig. SVM

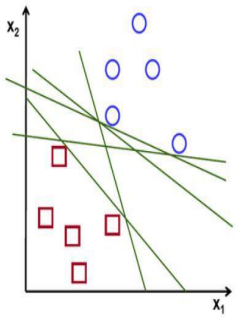

Fig. Hyperplane in SVM

The image above which has blue data and red data. As we know, in KNN, for a test data, the distance to all the training samples is measured and one with minimum distance is selected. The memory requirements to store all the training-samples are high and measuring all the distances takes a lot of time. The primary goal here is to describe a line that divides the data into two regions uniquely. This kind of data which can be divided into two with a straight line is called Linear Separable. In the above given image, numerous such lines are possible to detect. To avoid the noise in the incoming data, the line should be passing as far as possible from all the points and this data should not affect the accuracy of classification. Therefore, a line which is farthest from all data points will provide more immunity against noise. Hence, SVM tries to find a straight line (or hyperplane) with largest minimum distance to the training data samples.

The training data to expected to find this decision boundary. In the image above, the training data are the shapes filled up with color. This training data is support vector and the lines passing through them Support Planes.

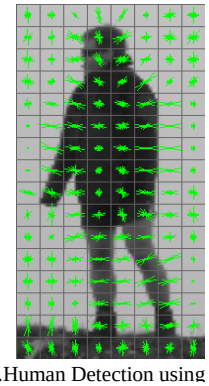

Fig .Human Detection using SVM

A. *Algorithm*

For this algorithm, we will use the CIFAR-100 dataset and extract train and test images for the classes baby, boy, girl, man, and woman. Each class consists of 500 train and 100 test images. We will split the data into train, val, and test sets. Also, a development set will be created which is a small subset of the training set. We also have put some sanity checks at some points in the algorithm implementation. We will then compute the image mean based on the training data and subtract the mean image from train and test data. To make the computations faster, we append the bias dimension of ones so that our SVM so that our algorithm can focus on optimizing a single weight matrix.

Fig. Raw Image (image seems distorted as the size of the image is very small i.e. 32 *32)

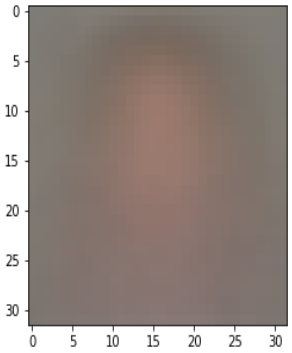

Fig. Mean Image

Post that, we evaluate the loss for the datapoints and generate a random SVM weight matrix of small numbers. There are several ways to define the details of the loss function and for our process, we will be using the in-built method. Once, we have all the required data, we start training our model with the training dataset. Then the model is fed with the test dataset gradually, and the accuracy is computed by checking false positive and false negative.

In the image below, we have shown the correlation between loss and iteration number. Looking at the results we can clearly state that as the number of iterations increase the loss value decreases significantly.

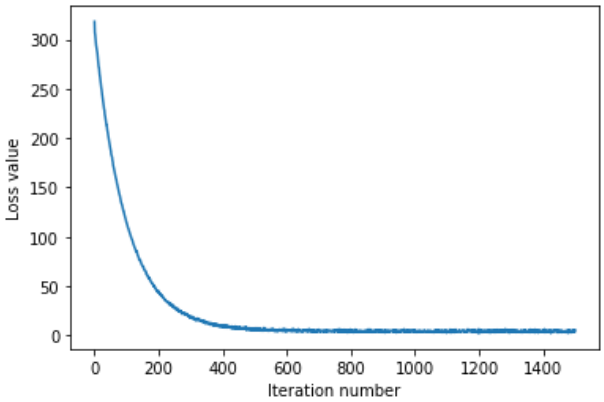

Fig. Shows the correlation between loss and the iteration number

B. *Performance*

Final results show that the accuracy is around 28% which is quite low. However, it might be because of the less training dataset. Hence, we added more data points to the training set and found that as the training data increases, the accuracy rate also increase by a small margin. Even with the increased accuracy, SVM seems a not so good fit for our problem statement to detect human in a digital image

Classifier	Training Set	Accuracy rate
SVM	500	28.32%
SVM	550	30.4%
SVM	580	31.03%

VI. COMPARING CNN, KNN AND SVM

After performing image classification using CNN, KNN and SVM we could see that CNN and KNN performed considerably better than SVM. KNN works better than SVM when there is a fixed data where it is divided into training data and test data but if we have to make a prediction model where we have to test on a real time samples which were no previously available then SVM will work better as it has a good learning approach. Mostly KNN performs better in static data. While CNN performed better than SVM as CNN extracts more features from the image data but SVM just maps the input data to some high dimensional space where it could reveal the difference between the classes. CNN are deep architectures while SVM is shallow.

Model	Accuracy
CNN	35%
KNN	37%
SVM	30%

VII. CONCLUSION

In this paper, we compared how well different models work in classifying types of humans. In our findings, k-nearest neighbor had the highest accuracy of 37%, convolutional neural networks had an accuracy of 35%, and support vector machine had an accuracy of 30%. However, it is determined that none of the models discussed have a high enough accuracy to be considered "efficient" in classifying types of humans.

VIII. FUTURE WORK

Detecting human beings accurately in a visual surveillance system is crucial for diverse application areas including abnormal

1. event detection
2. human gait characterization
3. congestion analysis
4. person identification
5. gender classification
6. fall detection for elderly people, etc.

REFERENCES

Basic format for books:

[1] "batchNormalizationLayer." Batch normalization layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html

[2] "ClassificationOutputLayer." Classification layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.classificationoutputlayer.html.

[3] "convolution2dLayer." 2-D convolutional layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html

[4] "Create Simple Deep Learning Network for Classification." MATLAB, https://www.mathworks.com/help/deeplearning/examples/create-simple-deep-learning-network-for-classification.html

[5] "dropoutLayer." Dropout layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.dropoutlayer.html

[6] "fullyConnectedLayer." Fully connected layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html

[7] "imageInputLayer." Image Input Layer - MATLAB, www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.imageinputlayer.html

[8] "maxPooling2dLayer." Max pooling layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.maxpooling2dlayer.html

[9] "reluLayer." Rectified Linear Unit (ReLU) layer - MATLAB, https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.relulayer.html

[10] KNN for Classification using Scikit-learn - https://www.kaggle.com/amolbhivarkar/knn-for-classification-using-scikit-learn/data

[11] sklearn.neighbors.KNeighborsClassifier - https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[12] sklearn.neighbors.DistanceMetric - https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

[13] Understanding SVMs': For Image Classification - https://medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700

[14] Large-scale Image Classification: Fast Feature Extraction and SVM Training - http://rogerioferis.com/VisualRecognitionAndSearch2014/material/papers/SuperVectorCVPR2011.pdf

[15] Comparison: KNN & SVM Algorithm - https://www.ijraset.com/fileserve.php?FID=11852