Grenze International Journal of Engineering and Technology, June Issue

A Framework of Image Processing Machine Learning Algorithms

Apoorva M[1], Mariyan Richard A[2], Amali Sunitha P[3] and Pushpavathi[4]
[1-4]Department of Computer Applications, Atria Institute of Technology, Autonomous, Bangalore, Karnataka, India
Email: apoorvam515@gmail.com, mariyanrich01@gmail.com, sunithaami32@gmail.com, pushpavathip1999@gmail.com

***Abstract*— Picture sorting sits at the core of smart machines. Nowadays computers learn to tell images apart using different math-based methods. Instead of typing words people drop photos into search boxes. Two well-known learning systems stand out - one uses neighbors, the other draws lines between groups. Machines must recognize scenes before they take action. Understanding visuals helps robots decide what comes next. Looking at a picture, the machine must ready itself to label what it sees inside. Humans find this task quite simple. Machines face more hurdles along the way. Processing one image takes multiple steps before any answer shows up. Results can point toward SVM outperforming KNN in certain cases.**
Classifiers. SVM Support Vector Machine. KNN K Nearest Neighbors.

I. INTRODUCTION

A photo gets sorted into set groups by what it shows - that is image classification. Though people find this easy, machines often struggle. Because of that challenge, some online tests show pictures to figure out if you are human. These puzzles rely on how hard images are for computers to understand. Fixing flaws in those puzzles matters for web safety, watching systems, sensors, and searching online. Creating smart tools that sort photos automatically has grown more urgent through time.Most times, sorting pictures into groups feels tricky when one type looks too much like another. Yet each group still holds many different forms within itself. Outdoors, sunlight shifts - morning light isn't noon light - and shadows stretch differently by season. These shifting lights blur the lines between what belongs where. On top of that, messy backdrops pull attention away from main subjects. Telling apart foreground from clutter becomes yet another layer before labels make sense.From time to time, images get broken down into pieces that match things you'd see in real life - that's what segmentation does. When it's fully done, each separate part clearly stands for one actual thing in the photo. Sometimes though, instead of clear matches, areas link only loosely to real items, maybe just by general type. These looser splits still divide the image but without sharp ties to specific visible objects.

II. KNN CLASSIFIER

From needing better ways to sort data without clear statistical models, KNN started taking shape. A 1951 US Air Force aviation medicine paper, not widely seen at first, introduced it. Hodges and Fix offered something different - no strict formulas, just measuring how close points are. Their method grouped items based on nearby examples already labeled. This early version leaned heavily on proximity, drawing conclusions from what surrounded each case. One label wins if most neighbors share it. By 1967, researchers dug deeper into its behavior.

Grenze ID: 01.GIJET.12.2.641_26

Certain limits were found - for example, how wrong the nearest neighbor choice could get. Over time, these insights helped clarify when and why the technique holds up.

Later came waves of study sparked by KNN's known traits - new ways to reject uncertain picks appeared, adjustments tuned to Bayes risk took shape, spacing methods gained weights, fuzzy shifts found footing. Recent efforts targeting classifier behavior mostly chase high and low limits when sorting gets hard, like jagged edges in class lines seen under Audiberth or Tsybakov and also Kohler plus Krzyzak. Mammens alongside Tsybakov opened room to slide smoothness along a scale, forming kind of link from slick setups we handle down into ragged ones.

The k-nearest neighbor method goes by the name KNN. One of the simplest tools in machine learning, it sees wide use across fields. Classification happens based on how close an item sits to others nearby - its class comes from those closest in line when counting up to k. When only the single closest point matters, that is k equals one, then it turns into just the nearest neighbor rule, where proximity decides everything. Distance plays a central role throughout this process [40]. Each item takes shape as a point, mapped using coordinates inside a space built from many features. When working in many dimensions, measuring how far apart two points are often relies on straight-line distance. This method shows up often in tools that classify data, especially one called KNN. Picture having a point you want to understand, plus a group of known examples nearby. Instead of guessing blindly, the system looks at the closest matches around it. From those neighbors, it picks the most common label among them. That choice becomes the predicted class for the original point. It does not invent new answers; it follows what the crowd says. Simple rules guide its decisions each time. For any unknown case, similarity decides the outcome. Closest wins. Labels follow proximity without extra assumptions. The process repeats every single time a prediction is needed.

The Basic Steps In KNN Classifiers

- Start by measuring how far the new data point sits from each earlier one. Those points? They’re already tucked into groups. Distance gets worked out using what’s been sorted before. Every past example plays a role. The method leans on existing cluster assignments without starting fresh.
- Start by arranging distances from shortest to longest, then pick the K points closest in measure. Finish with only those nearest matches included.
- A fresh data point lands in the biggest group when picking from K neighbors. Voting kicks in by counting those nearest picks, then assigning membership where most votes pile up.

A case to study how a KNN classifier works:

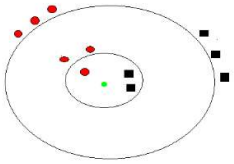

Fig 1: Illustration of KNN Classifier

When K equals three, the label turns black since two black squares show up within the inner ring, along with just a single red dot. Five neighbors shift it to red instead - three of those nearby shapes being red circles, while two remain black squares inside the wider boundary.

Because it looks at nearby data points, the KNN algorithm sorts items into categories. When using k-nearest neighbors, decisions happen right at prediction time instead of building a model early. Without needing to know much about how data spreads out, this approach stays basic yet functional. Classification works by keeping every training example and waiting until new input arrives. Then, whatever group appears most among the closest K matches becomes the answer. If only one point decides the outcome, that is called the nearest neighbor rule. With just one look around, each item gets labeled like its immediate peers do. Around here, similarity shapes identity through proximity. When a sample's category isn’t clear, looking at similar ones nearby might help decide where it fits. Starting from an unlabelled item and a group of known examples, measuring how far it sits from each reference point reveals possible matches. Closest match wins - that means the example with the tiniest gap guides the labeling decision. So assignment follows what the most similar past case was tagged as [30]. How well this method works hangs mostly on two things: which counting rule you pick for neighbors and how distance gets measured across items.

When k is tiny, guesses get shaky because there are too few nearby points, plus messy or wrong labels make it worse. To even things out, boosting K helps by pulling in more surrounding examples. But go too high and

everything blurs - distant oddballs sneak in, muddying the result. Picking just how many neighbors to include stands as the key challenge, shaping how well KNN works in practice. Tiny training sets often shake up how well KNN picks the right number of neighbors. When data is scarce, choosing K becomes shaky, which quickly drags down accuracy. This happens because small changes in input heavily sway the math behind picking K. Ready-made tools exist that handle these hiccups without extra effort. Each tweak in setup shows different results depending on the tool used.

A. K Nearest Neighbor Prediction

Once the value of K chosen, one can make prediction based on KNN examples for regression KNN prediction is the average of the KNN outcome.

$$Y=\frac{1}{K}\sum_{i=1}^{K} yi \quad \text{or} \quad z=\frac{1}{K}\sum_{i=1}^{K} zi$$

Every now and then, yi or zi stands for the ith item in the example set. Meanwhile, y or z reflects what the query point turns out to be. Unlike regression, classification tasks make KNN decide through a vote. The choice that wins gets assigned to the query. Each K counts just as much - even if one is far away while another sits close. It begins with picking larger values of K, though closer ones matter more. That shift happens by what people call distance weighting.

B. Distance Weighting

Close points often look alike, so their influence shapes what KNN predicts. That similarity guides decisions at the query spot. Weighting each neighbor differently adjusts how much they sway the result. A collection of values, called W, tunes this effect per nearby point. Distance matters because nearer examples get stronger voice through these assigned weights.

$$W(x, Pi) = \frac{\exp(-D(x,Pi))}{\sum_{i=1}^{K} \exp(-D(x,Pi))}$$

Where $D(x, p_i)$ is the distance between the query points x &i^{th} case of p_i, the example sample. It is showed that the weights defined in this fashion above will satisfy.

$$\sum_{i=1}^{K} W(x0, xi) = 1$$

Thus, for regression issues we have

$$y= \sum_{i=1}^{K} W(x0, xi)yi$$

III. SVM CLASSIFIER

It started back at COLT-92, when Boser, Guyon, and Vapnik rolled out SVM - slowly it climbed, backed by solid reasoning rooted in statistical learning theory. Performance? Sharp in practice, showing up strong across areas like bioinformatics, text handling, even spotting patterns in images[48]. People from many corners - stats, optimization crews, machine learners, those digging into neural nets or function spaces - all found reasons to dig in.

A handful of methods begin with an SVM classifier, building sparse models through just a few support vectors. Rather than settling on one fixed fit, these approaches stack approximations - each adding another vector. Step by step, they form layered classifiers, growing complexity only as needed. What emerges is not a single stage but several, each shaped by increasing detail.

Example for the analysis of SVM classifier

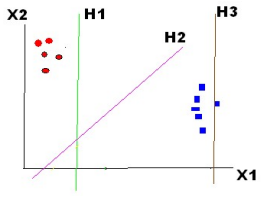

Figure 2 Example of an SVM Classifier

Some line - drawn through data points - fails to tell the two groups apart when it is brown. A green divider splits them, yet barely. One drawn in pink pushes separation as far as possible between sides. This fullest gap defines the best dividing line in the method. Dots sitting exactly where the edge touches get a special name. These examples shape the boundary just by being there.

Imagine a setup with two groups of points that can be split by a straight line. This dividing line works best when it keeps as much space as possible between both sides. Picture each group forming its own cluster. These clusters are made by linking all points from the same type together into solid shapes. The ideal separator sits right in the middle, far from both blobs. Distance matters here, because pushing the boundary away from either shape helps define a clearer gap. That wide gap becomes the strongest version of such a divider. Being distant from each compact form improves separation quality.

Right near the edge of the dividing space sit certain key points, known as support vectors. Each category brings at least one such point into play, sometimes several. From these select few, the best separating boundary gets fully shaped. Everything else in the data? It plays no role - toss them out, and the line stays put.

A line splits the two groups when there are just two features involved$F(x) = W_0 + W_1a_1 + W_2a_2$

Where, a_1, a_2 are attributes, W_0, W_1, $W_{2\ are}$ the weights.

The equation for the maximum margin hyper plane can also be give in terms of support vectors.

$$F(x) = b + \sum_{i=0}^{l} aiyi(xi.x)$$

Here's one way it goes: think of i as just a marker for a support point. Picture xi as the actual data tied to that spot, picked from the crowd. Now flip it - yi tells you what team that point belongs to, red or blue, really, either 1 or −1. Slide over - the pair x and y? That's someone new knocking at the door, waiting to be sorted.

Dotting x with xi gives value nineteen. Working out which points are key happens during analysis. The broadest gap between groups shows up through calculation. Solving it fits into common methods for tuning outcomes. Picking boundaries follows set patterns found in similar tasks.

Most times, picking a tiny k makes things shaky due to scarce data, also messy labels mess it up worse. Flip that - huge k brings its own trouble, pulling in too many odd points from far-off groups nearby. Balance sways wildly either way, stuck between gaps and glitches.

Because of how it works, KNN guesses a query's class based on nearby data points in space, using the distance to its Kth closest match. That guess changes sharply depending on what value of K you pick. Instead of just counting votes equally from neighbors, some approaches give stronger weight to closer ones - this helps when far-off points confuse the result. If data spreads unevenly across space, picking any fixed K gets tricky. Farther neighbors add stability, reducing random errors while blurring class lines less abruptly. Yet deciding one best K rarely works everywhere since each situation shapes the pattern differently.

One way to start is by looking at how machines learn from labeled data. Though often grouped together, methods differ sharply when tested. Instead of blending everything fast, some prefer checking each step slowly. Decision trees split choices like paths in woods. Rules-based systems rely on clear if-then chains built ahead of time. Perceptrons come in layers - some shallow, others deeper - but all turn inputs into decisions. Even simpler ones manage basic shapes in data; complex versions handle twists better. Radial networks measure distance from centers instead of following lines straight through. Bayes' rule guides certain models using probability as their compass. Others map cause and effect across linked nodes shaped like webs. Nearest neighbors judge by closeness, trusting

Somebody suggested a way to spot human-made things in sharp satellite images. This method uses SVM, guided by labeled examples, to tell what is shown. Instead of separate models, one general setup handles every kind of object. Shapes and patterns come first when sorting through stored samples. Features pulled out include basic ones - think Fourier-Mellin and Flusser's moments - and more complex traits too. Among those higher clues: how line directions spread out, where crossing points sit relative to the frame middle, shifts between region cores and image center, plus how tightly grouped each closed shape appears. These geometry-based details then feed into classifiers - one being "one versus rest," another testing categories head-to-head. Each version was checked on real cases after training. Another method used $N<(N-1)$ SVM, applying geometric traits to spot and identify human-made items. Efficiency came fast, accuracy stayed high, even when sample counts were low [11].

Out of two methods, KNN plus SVM were put together for guessing what a eukaryotic protein might be. Running at once, each handled the same data separately. One result came from KNN, another popped up from SVM. Merged together after, their answers got weighed through majority choice. The final call on the protein type emerged that way [15].

One study looked at how radar signals change when people move. Instead of using regular methods, it focused on sound-like patterns pulled from radio waves bouncing off humans. These patterns came from short bursts sent by two small radars watching different body motions. To tell falls apart from normal actions, the system turned those echoes into numbers called MFCCs. Rather than sticking to one method, it tested both a neighbor-based model and a boundary-drawing algorithm.

Using SVM for sorting SMS messages worked well when filtering features by how often words appeared. Words were picked based on their occurrence across messages, nothing more. A mix of real texts - 1002 pulled at random from two sources, NUS and Jon Stevenson - and 82 spam samples grabbed from a mobile junk archive made up the data pile. Running everything inside R kept things steady. Results turned out fine, no surprises there, just solid outcomes linked to that basic word count method [8].

TABLE I: SHOWS THE SURVEY ON VARIOUS PARAMETERS

Sl. No	Author(s)	Database	N0 of samples	Training set	Testing set	Accuracy	Algorithms
1	Qi, Gu and Zhifei song	African people, Flowers, Fashion	100	80	20	Accuracy rate high for flowers than other two	SVM, KNN, LR
2	Jinho Kim. Et.al	Airplane, cars, faces, motorbikes	3505	2800	700	87.24% 93.39%	KNN SVM
3	D.S. Guru, et.al	flowers	1250	20	30	90.13%	KNN
4	Yang song, et.al	COIL-20	20	-	-	-	GI-KNN
5	Subhransu Maji, et.al	Caltech 101	1360	15	15	52%	SVM
6	Y.Ireaneus, et.al	Mini-MIAS	75	-	-	88.75%	SVM
7	Fabrice Colas, et.al.	Text document	2000	1800	200	Good. Poor for SVM	Navie Bayes and KNN.SVM
8	Hao Zhang, et.al	CURet	92	-	-	-	SVM, KNN
9	Arash rezaei, et.al	AVC	1000	-	-	-	DT,SVM,KNN
10	L.R.Sudha, et.al	Human Gait	240	-	-	95.83% 97.92%	KNN SVM

IV. MERITS AND DEMERITS OF KNN

List of merits:

Easy to break down with numbers yet straightforward when putting it into practice

Few steps off perfect when things stay big yet uncomplicated

Because it relies on nearby data, adaptation stays sharp. Nearby inputs shape its responsiveness closely. From location-based details comes quick adjustment. Close-at-hand info drives flexible reactions. Its agility grows from immediate surroundings feeding change

Parallel implementation becomes very easy

List of demerits:

Storage requirements very large

Computationally intensive recall.

SVM Advantages and Disadvantages Compared to KNN

Putting the issue into a two-group sorting setup comes next. Classifying things in one of two ways shapes how it works. Splitting results into just two options changes the approach needed. A choice between only two outcomes guides the method used.

Learning it feels endless sometimes. Still, progress happens slowly. It demands patience every step of the way

V. CONCLUSION

Most times, SVM stands out among tools like KNN because it handles errors in training data without breaking down. Yet another strength lies in its speed during testing phases, even if setup gets tricky later on. Though built mainly for two-group sorting tasks, workarounds exist - like matching categories one against another - to stretch it into broader uses. These fixes tend to drag processing time down, simply put, they take longer. Domain-specific insights? Hard to feed them directly into SVM's logic, unlike some more flexible systems. When placed beside KNN, results often lean toward SVM’s favor, yet the edge isn’t guaranteed across every dataset type. Performance shifts heavily based on how messy, large, or scattered the input information happens to be. What ties certain data traits tightly to success in either method - that puzzle stays unsolved so far. Most times no single way of sorting things fits every situation perfectly. Problems keep showing up with how we currently classify

information. Figuring out the right approach often means testing many options before finding what performs well. Looking ahead, researchers might study patterns across various types of images to see why some methods succeed where others fail.

REFERENCES

[1] R szeliski, computer vision : "Algorithms and Applications", springer 2011.
[2] S.Savarese, J.Winn, and A. criminisi, "Discriminative object class models of appearance and shape by correlations" , proc. Of IEEE computer vision and pattern recognition, 2006
[3] A.Vedaldi and B. Fulkerson , VLFeat , "An open and portable library of computer vision algorithms",2008
[4] Bremner D. Demaine E.Erickson j,Iacono J. Langerman s, Morin P, Toussaint G, "Output sensitive algorithms for computing nearest neighbor decision boundaries", Discrete and computational geometry,2005.pp 593-604
[5] L. Fei-Fei, R.Fergus and P.Peron, "A Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories", IEEE computer vision pattern recognition 2004, workshop on generative model based vision 2004
[6] D.A Forsyth and J.Ponce, "Computer Vision, A Modern approach" , Prentice Hall, 2003
[7] Chris Ding and xiaofeng He, "K- means clustering via principal component analysis" , proceeding of international conference machine learning, july 2004.pp 225-232
[8] Parimala R. and Nallaswamy R. "A study on Analysis of SMS classification using document frequency threshold", I. J. Information Engineering & Electronic Business, 2014, No.1, PP. 44-50, 2012.
[9] Asha Gowda Kare Gowda, AsfiyaNasiha, Vayaram M. A. and Manuvanath A. S. "Exudates Detection in Retinal images using Back propagation neural networks" International Journal at Computer applications, Vol. 25, No. 3, PP. 25-31, 2011.
[10] Danny Roobaert and MarcVanHulle M. "View-based 3D object recognition with support vector Machines" in proc IEEE International Workshop on Neural Networks for signal Processing (NNSP99), Madison, USA, 1999.
[11] Jordi Inglada "Automatic recognition of man-made objects in high resolution optical remote sensing images by SVM classification of Geometric Image features". ISPRS Journal of photogrammetry & remote sensing Vol. 62, PP. 236-248, 2007.
[12] Kotsiantis S. B. "Supervised machine learning: A Review of classification techniques", Informatica, Vol. 31, No. 3, PP. 249-268, 2007.
[13] Liang Liu, Mihailpopescu, Marjoneskubic, Marilyn Rantz, Tasikyardibi and Paul Cudding "Automatic Fall detection based on Doppler Radar Motion signature", Biological conversation, Vol. 75, No. 1, PP. 222-225, 2011.
[14] Zhen Mei, Qishen and BaoxianXe "Hybridised KNN and SVM test gene expression data classification", Life science journal, Vol. 6, No. 1, PP. 66-66, 2009.
[15] Liqi Li, Hong Kuang, Yuan Zhang, Yue Zhov, Kaifa Wang and Ying Wan "Prediction of eukaryotic protein sub cellular multi localization with a combined KNN-SVM ensemble classifier", Journal of computational Biology and Bioinformatics Research, Vol. 3, No. 2, PP. 15-24, 2011.
[16] Jackson J. E. "A user's guide to principal components", John Wiley & sons, New York, 1991.
[17] Rafael Gonzalez C. and Richard Woods E. "Digital Image Processing", Pearson Education, 2nd Edition, 2003.
[18] Muharrem Mercimeck, KayhanGulez and TarkVeliMumeu "Real object recognition using moment invariants", Sodhana, Vol. 30, Part 6, PP. 1055-1059, 2004.
[19] Sudha L. R. and Bhavani R. "Performance comparision of SVM and KNN in automatic classification of Human Gact patterns", International Journal of computers, Vol. 6, No. 1, PP. 19-28, 2012.
[20] Te-Hsiu sun, Chi-Shuan Liu and Fang-Chin Tien "Invariant 2D object recognition using Eigen values of covariance matrices, re-sampling and auto correlation", Expert systems with applications, Vol. 35, PP. 1966-1977, 2008.
[21] Zhang H., Berg A. C., Maire M. and Malik JitendraEecs U. B. "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition", in proc. IEEE conference on computer vision and pattern recognition, Vol. 2, PP. 2126-2136, 2006.
[22] Bailey, T. & Jain, A (1978) " A note on distance – weighted K- Nearest Neighbor rules". IEEE Trans systems, Man, Cybernetics, 8:311-313.