A Comparison of Neural Networks and Support Vector Machines in Digits Classification

Aparna Surendra and Zara Thomas

HAndwritten digit recognition has received considerable attention from the machine learning community. The MNIST('Modified National Institute of Standards and Technology') is one of the leading datasets for benchmarking classification algorithms [7], and contains tens of thousands of handwritten digits. Since 1999, the MNIST database continues to be a reliable resource for researchers as new machine learning techniques emerge. While digit recognition may, in 2018, seem a trivial task, there is a reason that the MNIST dataset remains evergreen: handwritten digit recognition is an activity that encompasses many of the functions required to create intelligent behaviour.
The main aim of this study is to critically compare and contrast the performance of two families of machine learning algorithms: neural networks, and Support Vector Machines (SVM) to recognise handwritten digits — a multiclass classification problem.

I. Overview of Dataset

We use modified version of the original MNIST dataset, devised for a Kaggle competition. It has 40,000 observations (compared to the 60,000 observations in the original dataset), of 10 handwritten digits from 0 to 9. The observations are roughly balanced across classes, with a range of 3795 to 4684 observations per class. There is a median of 4157 observations per class, and the mean is 4200.
MNIST is a popular dataset because most of the pre-processing has been done by the original authors. The images have been normalised to fit in a 20x20 pixel box (while preserving their aspect ratio) and the images have been centered in a 28x28 image[3]. For the purpose of this study, we do not conduct any further pre-processing. However, pre-processing steps such as centering the digit based on bounding box (instead of center of mass) have been shown to improve accuracy for SVM, and deskewing (shifting the lines of the digit to make it vertical) has been shown to improve the accuracy of 1 and 2-layer neural nets [3]. This could be an opportunity for further work.

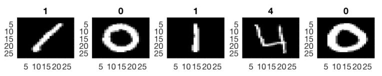


Fig. 1: Sample images from the dataset

II. Machine Learning Methods

A. Support Vector Machine (SVM)

Support Vector Machine is a supervised classification algorithm. Given a set of training examples, SVMs construct a hyper-plane that maximally separates the positive and negative instances in the training set. The resulting hyper-plane only depends on the data points that lie closest to the decision surface, termed the 'support vectors'. In the digit classification problem, each pixel in a 28x28 image is represented in its' own dimension i.e. there are 784 dimensions in total. The hyperplane must divide the data points into two classes in this 784 dimensional space. The equation of the hyperplane separating two different classes is given by the relation where w represents the weight vector and b a bias threshold:

$$f(x) = w^T x + b$$

In very high dimensional problems such as digit classification, the data is not always linearly separable. The optimal solution is one that separates the bulk of the data and ignores any noisy images. The standard approach is to allow some errors in the decision margin, i.e. outliers are on the wrong side of the margin. Soft-margin SVMs minimise the training error by trading off the margin size and ensuring that the data lies on the correct side of the margin [9]. Increasing the parameter C places more weight on slack variables, so that the optimisation attempts to make a stricter separation between classes. Non-linear classifiers can also be easily implemented by replacing the dot product with a non-linear kernel function. This enables the data to be transformed into a higher dimensional space without computing the exact transformation of the data.

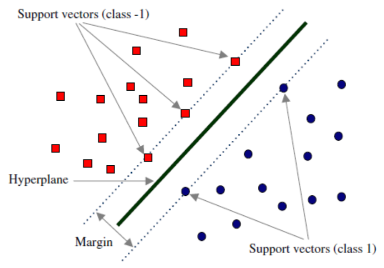


Fig. 2: Hyper-plane with maximal distance to positive and negative classes in SVM [10]

As this is a multiclass classification problem, multiclass SVMs must be used to reduce the problem to a binary classification one. The two principal techniques in doing this are: 1) one-versus-all (OVA) and 2) one-versus-one (OVO). In OVA, one class is positive and the rest are negative for each SVM. This method exhausts all methods of positive class assignments. In OVO, one class is positive, another is negative, and the rest of the classes are ignored in each SVM. This method exhausts all combinations of class-pair assignments.

B. Neural Networks

A multilayer perceptron is a class of feedforward artificial neural network. The neurons are arranged in layers including the input layer(the set of input nodes), the hidden layers and the output layer. In classifying digits, the input layer is comprised of the pixel intensities in each of the digit images i.e. a 28 x 28 image will contain 784 input neurons. The activation of these units are propagated throughout the network, where each node in one layer connects with a certain weight w_{ij} to each node in the following layer. The classification of digits is then determined by the value of the output unit(s). If the output does not match the expect target value, an error signal is back propagated through the network. The connecting weights are adjusted to minimise or eliminate the error. Within the neural net family, it is now more standard to use convolutional neural networks (CNNs) for image classification tasks. As such, we include a simple CNN to complete our analysis. In an MLP, each unit within the first hidden layers consists of 784 weights. With 1000 neurons in a hidden layer of an MLP, this equates to 784,000 weights (also known as parameters) in the hidden layer. A CNN computes a convolved image (using a convolutional kernel), and reduces the number of parameters involved. CNNs have been considered state-of-the-art since 2012, when AlexNet was used to win the ImageNet challenge [1].

C. Comparison of Methods

We do a brief comparison of the two methods here:

SVMs	Neural Networks
SVMs can take a longer time to train large or complex dataset as they must create a hyperplane that can separate all classes.	MLPs are less affected by the size of the dataset (this may affect the time taken to backpropagate but the parameters are not determined by the number of data points present).
SVMs have shown to peform better than MLPs on the MNIST Dataset.	CNNs have performed better than SVMs on the MNIST dataset. The convolution kernels provide a way to reduce dimensionality while preserving some spatial relationship data. This allows them to work especially well on image data.
SVM performance can be significantly affected by kernel selection.	MLPs with a large number of free parameters can be prone to overfitting. This can be overcome by methods such as early stopping.

D. Hypotheses

Based on the literature, we have several hypotheses:
1) We expect SVMs to perform better than MLPs, as shown in multiple studies in object recognition [8] [3]. SVMs show great generalizability in constructing a decision surface that is able to separate a high number of classes without overfitting.
2) We expect the CNN to perform better than the MLPs, but to be slower. This is because convolution operators are more complex than those in fully-connected layers, even though CNNs involve fewer parameters than MLPs [3].
3) We expect the CNN to perform better than the SVM. [3]. We also expect the CNN to perform more slowly than the SVM. Methods like LeNet-5 have shown to require more operations in comparison to kernel SVMs [7]. However, our CNN is less complex than the LeNet-5, and may have less accuracy and faster performance.

III. Experimental Methods

Using MATLAB, we used the holdout method, partitioning the data into the training, validation and test set (at 56%,14% and 30% respectively). Using the holdout method enables the evaluation of how well the model will work in practice i.e. whether or not the model is generalisable to new data. For both MLP and SVM, we used Bayesian optimisation (over 30 iterations) to select the optimal machine learning hyperparameters.
'Tuning [hyperparameters] is often a 'black art' requiring expert experience, rules of thumb, or sometimes bruteforce search' [6]. Bayesian optimisation approaches hyperparameter tuning more mathematically. In Bayesian optimisation, the algorithm's performance is modelled as a sample from a Gaussian process. This allows the estimation of the hyperparameter values with the highest probability of further minimising the objective function. In using Bayesian Optimization for both SVMs and Neural Networks, hyperparameters are found which minimised the five-fold cross validation loss within the models.

A. SVM

We apply three different kernal types to the SVM: linear, gaussian and polynomial kernel (2^{nd} and 5^{th} order). We then apply Bayesian optimisation to two hyperparameters — the box constraint parameter (C) and the kernel size (σ).

- *Box Constraint: [1e-4 1e4]*

As mentioned in section I, the box constraint parameter determines how strict the separation between classes is in the model.

- *Kernel Size: [1e-4 1e4]*

In MATLAB, the kernel size is an additional parameter used to tune SVM models. In gaussian kernels, the kernel size is equivalent to setting the σ parameter as shown in the equation below:

$$k(x, x_i) = exp(-\frac{\parallel x - x_i \parallel^2}{2\sigma^2})$$

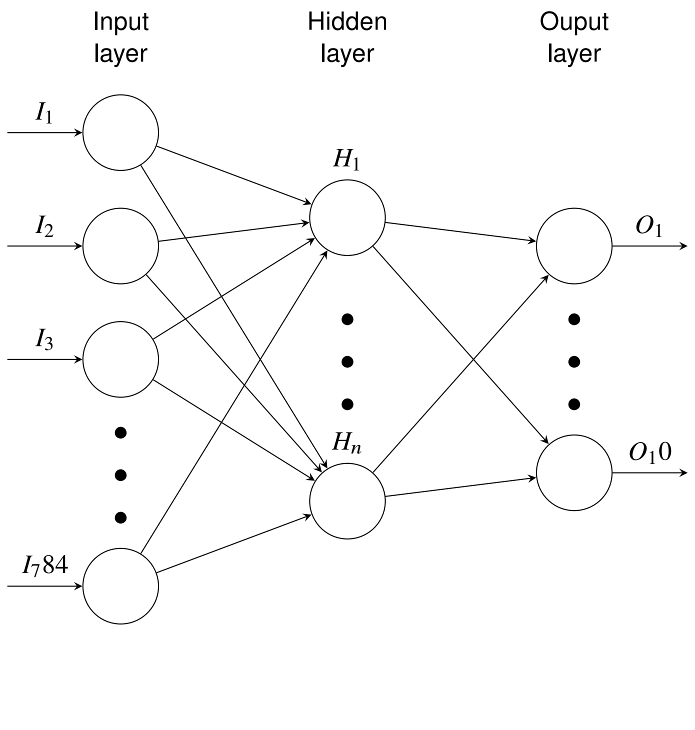
σ plays a role in amplifying the distance between x an y. However, in polynomial and linear kernels, the kernel size is a scaling factor applied to each of the training inputs as follows (where d is the polynomial order):

$$k(x, x_i) = (x \cdot x_i)^d$$

B. Neural Networks

We evaluate two types of neural networks: two 2-layer MLPs, and one convolutional neural network (CNN). The 2-layer MLP architectures and hyperparameter ranges were selected based on best practices in the existing literature, while the CNN was designed as a point-of-comparison (when coupled with Bayesian optimisation, the state-of-the-art CNNs would have been too computationally expensive for this project).

- *Two-Layer Multilayer Perceptrons*



The 2-layer MLPs have an input layer of 784 units (a unit for each pixel) and an output layer of 10 units (a unit for each possible class). We experiment with two types of hidden layers: $H_n = 800$ and $H_n = 1000$.
We use a ReLu activation function for layers I and H, and a softmax activation function for layer O, which provides normalised probabilities for each class. To prevent overfitting, training stops automatically (early stopping) if validation loss increases.

- *Convolutional Neural Network*

Our CNN architecture has an input layer, two hidden layers: a convolutional layer with a $5x5$ kernel (with ReLu activation) and a max-pooling layer, and an output layer: a fully-connected layer with 10 units (with softmax activation).

For each neural network, we use Bayesian optimisation across three hyperparameters in the following ranges:

- *Momentum: [0.8 0.95]*

We use stochastic gradient descent with momentum, which improves the speed of learning. As Rumelhart et al. write, "The momentum term helps average out the oscillation along the short axis while at the same time adds up contributions along the long axis" [4].

- *Initial Learning Rate: [1e-3 5e-2]*

We use time-based learning rate decay, to help with convergence, where (α) denotes the learning rate.

$$\alpha = \frac{\alpha_{initial}}{k^t}$$

We set $k = 10$ and $t = 1$ epoch, but vary $\alpha_{initial}$.

- *L2 Regularization: [1e-10 1e-2]*

Regularisation penalises model complexity and reduces overfitting. L2 regularisation penalises the squared magnitude of all weights in the cost function by λ, the regularisation strength.

IV. Findings and Analysis

We evaluated the best-performing iteration (of the 30 iterations conducted with Bayesian optimisation) for both SVM and Neural Networks experiments.

A. SVM

SVMs required Bayesian optimisation whenever the kernel function was modified i.e. for linear, gaussian and the polynomial kernels. Although both OVA and OVO models were considered in optimising the hyperparameters, OVO models achieved lower training error rates in comparison to OVA models. Hence, the results in table 1 are all based on OVO models. However, it was found that Bayesian optimisation was very inefficient in fine-tuning the hyper parameters. Hyperparameters found for the polynomial and gaussian models gave very high test error rates (approximately 80%). Thus we had to use a grid search method to find optimal parameters using a process of elimination.

Model	Test Error Rate	Runtime	Optimal Parameters
Linear	0.0571	46.439s	$\sigma = 29.367$ $C = 0.0010607$
Gaussian	0.0250	33s	$\sigma = 2500$ $C = 75$
Polynomial ($2°$)	0.0259	28.48s	$\sigma = 500$ $C = 128$
Polynomial ($5°$)	0.0320	325.31s	$\sigma = 1000$ $C = 0.00128$

TABLE I: Best-Performing Iterations for SVMs

The Linear SVM classifier obtained the poorest result of all the SVM models, with a test error of 5.71%. However, this is lower than the 15.38% error observed with a similar model by Maji et al., a difference that may be explained by the fact that they only trained on 1000 samples[7]. The SVM needs to see more data to be accurate.
The gaussian SVM achieved the lowest test error (2.5%), which corresponds to the literature— gaussian models perform the best among the SVM models [3]. However, our gaussian model error rates are slightly higher than those reported by LeCun's model (1.4%).
The 5^{th} order polynomial model achieved slightly lower test error (3.2%) than those observed by Schoelkopf (3.9%) [8], and the 2^{nd} order polynomial model achieved even lower test error (2.6%) . However, this is the inverse trend to those identified by Schoelkopf, where 5^{th} order polynomial model performed better than the 2^{nd} order. This could be because of the difference in datasets (Schoelkopf used the USPS digit datset). Our findings do correspond to Schoelkopk's in one respect: they note that the choice of kernel function is not imperative in achieving good generalisability. In fact, our non-linear models have very similar error rates. And, in training the models, it was also noted that changes in the parameter C did not have a significant impact on the error rates, as observed in literature [11].

B. Neural Networks

Model	Test Error Rate	Runtime	Optimal Parameters
MLP (800 units)	0.0447	13.147s	$\alpha_i = 0.0043126$ $m = 0.80083$ $\lambda = 1.7298e-10$
MLP (1000 units)	0.0445	20.327s	$\alpha_i = 0.0024931$ $m = 0.84564$ $\lambda = 4.324e-05$
CNN	0.0314	325.31s	$\alpha_i = 0.0010967$ $m = 0.94801$ $\lambda = 5.3699e-10$

TABLE II: Best-Performing Iterations for Neural Networks

The MLP $H_n = 1000$ has an error rate of 0.0445 (a 0.0005 improvement on a similar model described by LeCun et al. in 1998 [2]). owever, the MLP $H_n = 800$ has an error rate of 0.0447, which is much higher than the best-reported result of 0.016 (by Simard et al.) [5]. When reviewing their paper, it appears that our model is similar in key respects — use of cross-entropy as a loss function, no additional pre-processing of the data. However, Simard et al. do not use momentum, and have a slightly different λ decay rate. As such, we should modify these hyperparameters to exactly replicate their study. Our results suggest that adding units to the MLP's hidden layer have a negligible performance boost when compared to the additonal runtime (MLP $H_n = 800$ has a runtime of 13.15s compared to $H_n = 1000$'s 20.3s). This would be further compounded if we were able to replicate Simard et al.'s paper, as $H_n = 800$ would — in fact— perform much better.
As expected, the CNN performs better than the MLPs, but is slower (~16x slower than MLP $H_n = 1000$; ~23x slower than MLP $H_n = 800$). The CNN does not perform as well as the best-reported results, which range from 0.017 to 0.0053 for data without pre-processing[3]. This is not surprising, as we devised a simple CNN for this project. However, our findings suggest that even the simplest CNN can outperform an MLP.

C. Comparison

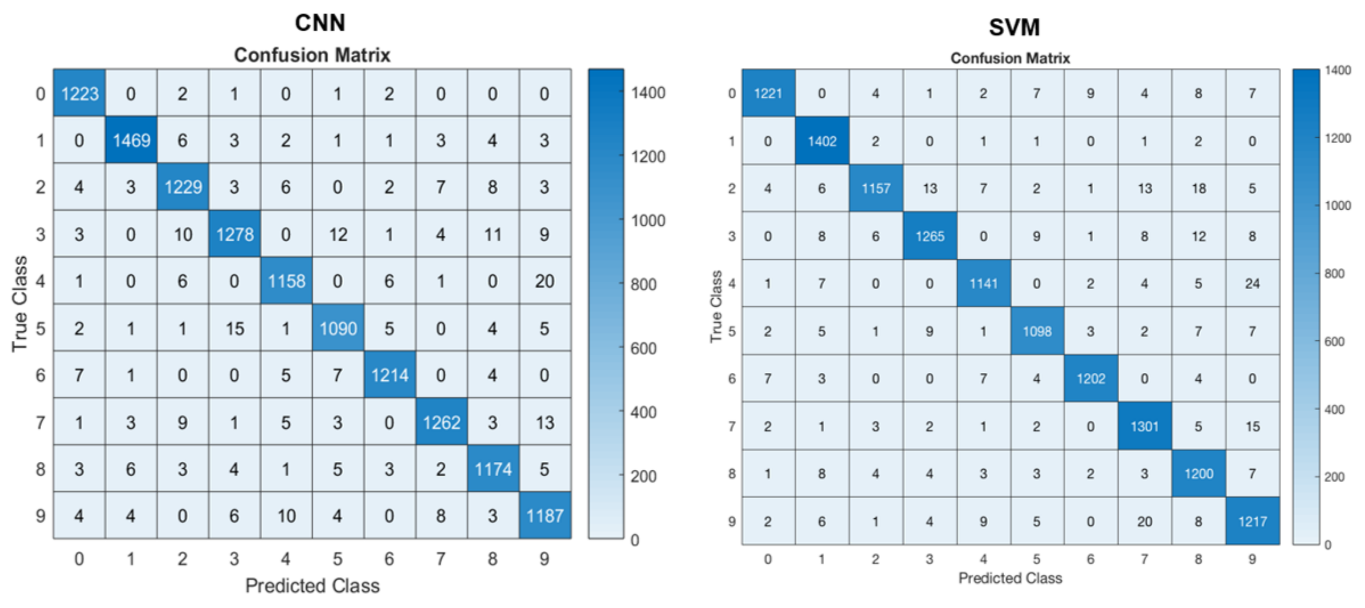


Fig. 3: Confusion Matrices - CNN and SVM

Run-time
As expected, the SVMs have a longer run-time than all the MLPs. Even when using a GPU to run the SVMs and a CPU to run the MLPs, there was a marked difference in run-time: the fastest-running SVM (Polynomial, 2^0) took 28.48s on a GPU, compared to the fastest-running MLP (800 units) at 13.147s on a CPU. Interestingly, the run-time for the most complex (and by extension, the longest-running) SVM on a GPU was comparable to that of the CNN on a CPU, at around 325.31s. CNNs have a more complex architecture than MLPs, but should be comparable (or even slower)

than SVMs. This result is most likely explained by the fact that the CNN used in this paper was quite simple. More complex CNN architectures such as Le-Net 5 may have longer run-time.

Accuracy

The linear SVM performed the worst among all models tested however, the non-linear SVM models surpassed both MLPs tested with significantly lower test error rates (2%). The CNN was only able to narrowly beat the 5^0 polynomial by 0.1% - surpassed by both the gaussian and 2^0 polynomial models. In comparing the confusion matrices for the best models for both SVMs and Neural Networks (gaussian and CNN respectively), the CNN was able to predict 4/10 of the digit classes more accurately than the gaussian SVM (5,7,8 and 9). Lower accuracies for these digits were also observed by Hartwick when using gaussian SVMs [11].

Testing Hyperparameters

Identifying optimal parameters in SVMs was a real challenge in creating a good classifier. This challenge was amplified by the high computational processing required due to the high number of dimensions within the images. Although, good results were obtained from using bayesian optimisation for neural networks, the experience with SVMs suggests that better hyperparameters may be obtained by using the grid search technique.

V. Conclusions and Future Work

Our findings suggest that the simplest CNN is able to outperform a MLP but not the non-linear SVMs. As our simple CNN outperformed the other models, further work should develop a more sophisticated CNN architecture with multiple convolutional and/or pooling layers. Within the SVM family, non-linear kernals far outperformed their linear counterparts, and future work should focus on this variation.

Looking ahead, research has demonstrated that the two methods can be combined: a CNN architecture with a linear SVM as a final layer (replacing the SoftMax)[12]. As one option, we could analyse a variation of this, using the non-linear kernel SVM as the final layer.

Across all methods, elastic distortions of the data can improve performance. For instance, the best performance of CNNs with elastic distortions range from 0.008 to 0.0035 (compared to 0.017 — 0.0053 without); for SVMs, as low as 0.0056; and for 2-layer MLPs, as low as 0.007 [3]. This suggests that future work should include use of elastic distortions.

References

[1] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (pp. 1097-1105)*.

[2] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition.*Proceedings of the IEEE, 86(11), pp.2278-2324*.

[3] LeCun, Y., Cortes C., Burges, C., *The MNIST Database of Handwritten Digits [Online]*. Available: http://yann.lecun.com/exdb/mnist [Accessed March 15, 2018.]

[4] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1985. *Learning internal representations by error propagation (No. ICS-8506)*. California Univ San Diego La Jolla Inst for Cognitive Science.

[5] Simard, P.Y., Steinkraus, D. and Platt, J.C., 2003, August. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR (Vol. 3, pp. 958-962)*.

[6] Snoek, J., Larochelle, H. and Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems (pp. 2951-2959)*.

[7] S. Maji and J. Malik,2009, Fast and Accurate Digit Classification, Eecs.Berkeley.Edu

[8] B. Schoelkopf, C. Burges, and a. Smola.,1999, Support Vector Learning, Adv. Kernel Methods, no. September, *pp.2-5*

[9] C. D. Manning and P. Raghavan,2009, An Introduction to Information Retrieval, in Online,*p. 1.*

[10] Amit, Support Vector Machines.[Online]. Available: https://amitranga.wordpress.com/machine-learning/support-vector-machines/, *[Accessed: 01-Jan-2018]*

[11] N. Hartwick,2015, Reproducing Results of Guassian Kernel SVM classifers on the MNIST Dataset

[12] A. F. Agarap,2017, An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification, arXiv1712.03541 [cs, stat]